SEXTANT

$2.95
March-April 1986
Issue No. 21

New: The Eight-Bit World

The Independent Magazine for Users of Heath/Zenith Microcomputers

EVERYTHING YOU ALWAYS WANTED TO KNOW ABOUT THE 8087 . . . BUT DIDN'T KNOW WHO TO ASK

See Seven Sorts in Action

How to Use PROMPT on the '100 and '150

A Quick Look at the Z200

PLUS: BASIC Questions and Answers • C Notes • Z100 Notebook

SEXTANT

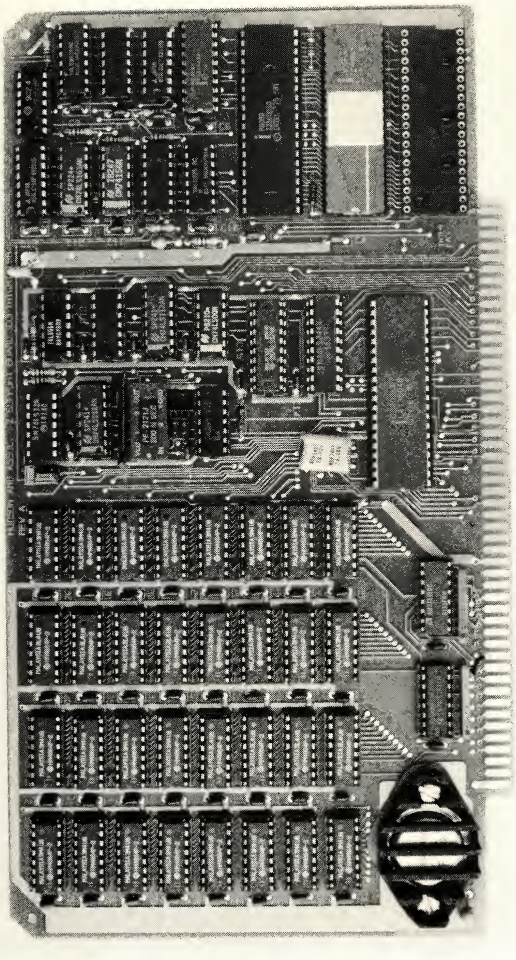


Photo courtesy of Alan T. Moffet

33

The 8087 speeds up operations that use floating-point numbers—by a factor of 20, in some cases. When a program wants to do a floating-point operation, the 8087 steps in and does the job faster and more efficiently. To get this performance, Z150 owners need only add the chip itself; Z100 users have a choice of three boards to hold the chip. And there's *lots* of software to take advantage of it.

Features

Issue #21
March–April 1986

Departments

Photo courtesy of Zenith Data Systems

The Editorial Eye

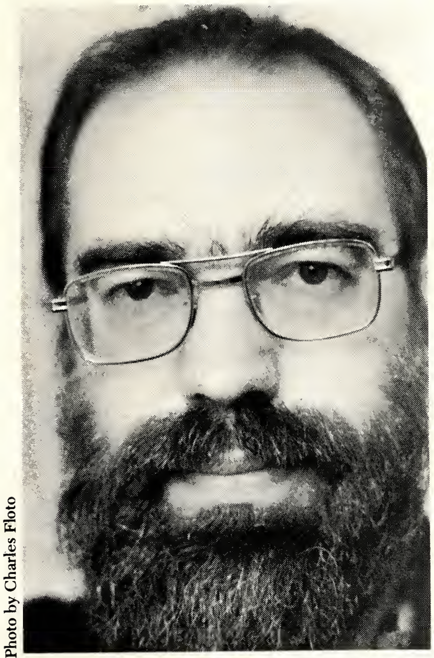


Photo by Charles Floto

About half the articles and columns in this issue contain information relating to the Z150 series of computers compatible with the IBM Personal Computer. Much of this information springs from their similarities to the Z100 series of dual-processor computers: the use of MS-DOS, Microsoft BASIC, and the 8087 numeric co-processor. In the future, we hope to give more attention to the aspects of the Z150 which distinguish it from other IBM-PC clones. These include differences in hardware and in monitor ROM firmware routines.

We have no intention of duplicating the coverage of subjects taken up in magazines published for IBM Personal Computer owners. There would be little advantage for anyone in our doing so. While there are only six issues of *Sextant* per year, magazines for IBM users publish a total of over 100 issues annually.

Thus, we're interested in articles about products for the Z150 which are either compatible with the Z100 or incompatible with the IBM Personal Computer. We're more interested in Zenith's innovations in hardware or software design than in their adherence to decisions made by other vendors.

Sometimes it seems as if most of Heath/Zenith's innovations are behind them. The H19 terminal and H89 computer provided 80 columns of upper- *and* lowercase characters years before Apple or Radio Shack products achieved such a professional display. Publication of the source code for the Heath Disk Operating System was a milestone not yet equalled by any other major computer vendor. HDOS itself went beyond CP/M in dating files displayed in the directory and allowing support for additional hardware through device drivers. (The latter was only equalled in version *two* of MS-DOS.) HDOS also includes utilities for testing disk drives and the floppy disks themselves.

As part of our continuing support for the innovative H8, H/Z89-90, H19, and HDOS, we're happy to introduce a new column in this issue: "The Eight-Bit World." It's conducted by Walter J. Janowski.

You may recall my mentioning last issue that I'd attempted to take pictures at CHUGCON without loading the camera first. Since then, I've purchased a camera with a film counter which accurately reflects the presence of film. The first picture I took with it is shown above. It illustrates the automatic focussing of the Minolta Maxxum held at arm's length from my face.

While we don't have any photographs of CHUGCON, Vickie Saxon has given us a picture of the conference better than I could have captured even if I had been using film.

Charley Floto

SEXTANT®

Charles Floto
Editor
Publisher

Jennie D. Allen
Managing Editor

John Walker
Technical Editor

John Gill
Production Manager
Subscription/Retail Service Manager

Bryan M. Rutberg
Advertising Manager

Gabriele Burkhard
Circulation Assistant
Production Assistant

Victoria Saxon
Editorial Assistant
Production Assistant

Elizabeth Saxon-Giles
Programmer
Production Assistant

Edward A. Byrnes
Contributor

Walter J. Janowski
Contributor

Joseph Katz
Contributor

Alison Phillips
Contributor

Subscription Service • 202/544-0900
Advertising Inquiries • 202/544-2868
Authors' Inquiries • 202/544-2867
Charles Floto • 202/544-0484

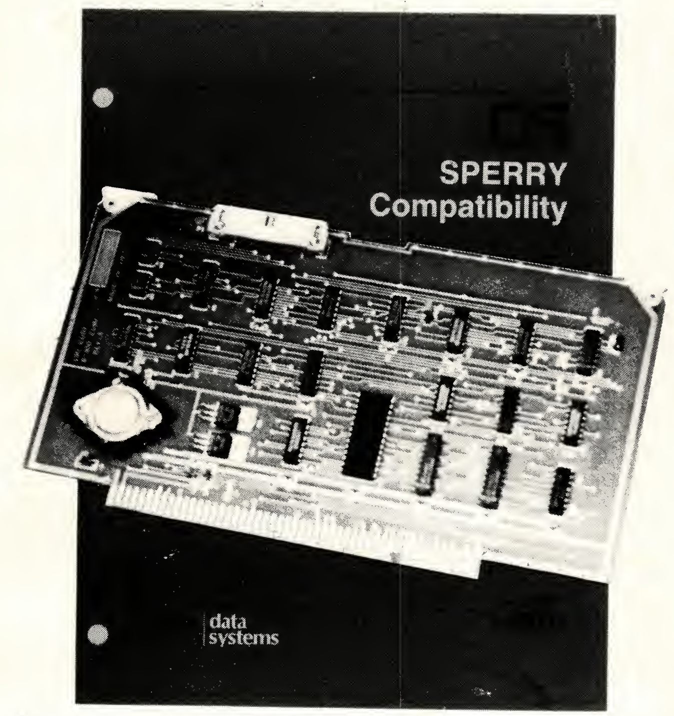
Sextant (ISSN 0731-2180) is published bimonthly by Sextant Publishing Co., 716 E Street, S.E., Washington, DC 20003. Second-class postage paid at Washington, DC and at additional mailing offices. POSTMASTER: Send address changes to Sextant, 716 E Street, S.E., Washington, DC 20003.

Subscription rates: 6 issues (1 year) in U.S.—$14.97; in Canada and Mexico—$17.25; overseas—$21.00. Payment accepted by check in U.S. dollars payable on a U.S. bank, by U.S. and international postal money order, and by Visa and MasterCard. Please allow six to eight weeks for delivery.

If You Have a Zenith Computer, Don't Buy a Sperry Terminal

Emulate Any Sperry Terminal

With the Chi Sperry/UTS Emulator, your Zenith computer can completely replace the Sperry UTS 40, 20, 400, Uniscope 200 or 100 terminals. With the Chi Emulator for the Zenith Z-100 computer (or the version for the IBM compatibles), you don't need a separate expensive terminal for each application. You save room on your desk and you save your budget, too.



The Chi Sperry/UTS Emulator allows your Z-100 to emulate the whole family of Sperry terminals. There is also a version for IBM compatible computers.

On the Contract

Getting a Chi Emulator for your computer is easy. Just call Chi and order the version you need. For the military, order from Contract F19630-83-D-0005, CLIN 0046AB, Model CC-463-1 Sperry UTS 40 Emulator for the Zenith Z-100, or order from Contract F19630-84-D-0009, CLIN 0025AE, Model CC-5063-1 Sperry UTS 40 Emulator option for the Tempest PC computer.

More Than Just Compatible

Transfer files back and forth between your Zenith computer and Sperry mainframe, use your own specially developed software to access the emulator for secure applications, or change the terminal type for different applications. You can do all of this and much more with the Chi Emulator.

Install, Configure, Emulate

Plug the Chi Emulator board into an open slot, enter a few simple Control Page commands to configure the type of terminal, and you are ready to begin productive use of your computer as a terminal.

Order Now

If you need a Sperry terminal, order the Chi Sperry/UTS Emulator instead. For just a few dollars, you will get full Sperry compatibility, access to your Zenith computer, more room on your desk, and greater productivity. *There is even a version for IBM compatible computers.* Order your Chi Emulator now!

Chi

OTHERS LISTEN. WE RESPOND.

26055 Emery Road, Cleveland, Ohio 44128
Telex: WUTCO TLX1 53-5737 Telephone: (216) 831-2622

Circle #104 on Reader Service Card

Letters

Increase MDISK's capacity

I have gotten numerous good suggestions from D.C. Shoemaker's articles. They are great.

In issue #19, he discusses RAM drives. I have a further suggestion for him from the Heath Users' Group's *REMark* magazine.

I have an H100 low-profile computer with 768K of memory. The MS-DOS version 2 operating system has a RAM drive utility, MDISK.DVD, with 64K capacity. I installed it, but 64K was not enough capacity. I like to save color screens, which take almost 192K disk space.

In the *REMark* magazine for March 1985 (vol. 6, #3), page 52, Pat Swayne has an article titled "Using the MS-DOS Memory Disk." In this he includes a chart showing how to change the MDISK.DVD utility to have a capacity of up to 512K. He also shows how to use DEBUG to make the changes. It works very well.

With this RAM drive of 512K capacity installed using MS-DOS, I still have 214K of RAM available.

I use AUTOEXEC.BAT to copy all the BASIC programs on the disk I am booting from to the pseudo-disk. I also use it with PeachText and PeachCalc. The increased speed over using a standard disk drive is certainly amazing.

Milton P. Klintworth
Drayton Plains, MI

Different syntax for TREECOPY

In Alison Phillips' "Z100 Notebook" column that appeared in your January-February 1986 issue, he listed a command which would copy all directories and files from one drive to another.

His command was:

TREECOPY C: | D: |

This command will not work for me. (I receive a SYNTAX ERROR response.) With my version (dated 4-04-84), the following command works:

TREECOPY C:*.* D:

You do have to make sure that you are in the root directory to transfer all directories and files.

Jim Woodruff
Milwaukie, OR

Let HDOS programs access the time

I am responding to Al Heigl's question in *Sextant* #20, on letting user-written programs access the time in Charles Horn's software clock. (See "A Software Clock for HDOS," *Sextant* #19.)

Listing A is an MBASIC program to find the time buffer in any version of the clock. The program works by finding the colons in the TEMPBUF label of the clock. Line 50 does this with PEEK statements. Line 70 checks for the two colons in the TIMEBUF label of the clock. The rest of the program looks into the clock, puts the time in TIME$, and prints the upper and lower address and the time for us.

Listing B is an MBASIC subroutine that looks into the clock buffer and places the current time in TIME$ for use by the user-written program.

Robert Phillips
Charlotte, MI

Clarifications and updates on Perks

I want to thank you for the coverage of my Perks Desktop Utility in your article on desktop utilities for the Z100 computer (January-February 1986).

The copy of Perks supplied for the review was 1.07; we are now (December 1985) shipping 1.15, and by the time this is published (March-April) we expect to be shipping 2.X. The following should be noted as a consequence of the differences between the current and reviewed versions:

The harmless bug observed in the Notepad feature (garbage characters sometimes appearing in the Notepad window) has been corrected.

The harmless bug in the Timer feature (failure to clear the screen under some conditions, making it hard to read) has also been corrected.

The compatibility of Perks with Lotus 1-2-3 was listed as only Partial in the article (although this was the highest rating received by any of the five products tested with 1-2-3). In fact, Perks works perfectly with Lotus 1-2-3, although you cannot use the Calculator export function. Since 1-2-3 can do its own arithmetic, we do not view this as being very significant. Please note, however, that you *can* use the notepad import function with Lotus 1-2-3.

In addition, Perks works just fine with the current version of Steve Robbins' WatchWord word processor—in fact, Steve was one of my earliest customers, and we are aware of no problems between Perks and WatchWord in either the current or previous versions. If anyone is having trouble using Perks with WatchWord, they should verify that they have current releases of *both* products, and that the "problem" is a true compatibility problem, and not simply a lack of sufficient memory to load the DOS, background utility, and the foreground program all at once. Perks and WatchWord work fine together, but you do need more than 192K of memory.

We are also unaware of any problems using the current version of Perks with dBASE II, although some problems were reported with versions of Perks prior to 1.13.

Although not mentioned in the article, I might add that Perks works with the current versions of Newline's PTP, Hilgraeve's ACCESS, KEA Systems' Z-STEM, and The Software Toolworks' PIE, among others. In each of these cases there were problems in earlier versions of these products which prevented them from working with *any* of the background desktop utilities (including but not limit-

```
10 '******FINDTIME.BAS*******
20 'TO FIND TIME BUFFER IN CLOCKS BY CHARLES HORN
30 DEFINT A-Z
40 FOR I = 4096 TO 5119                          'Memory address of Clock
50 IF PEEK(I)=58 AND PEEK(I+3)=58 THEN GOSUB 70     'Look for 2 colons
60 NEXT
70 IF PEEK(I-1)=72 THEN RETURN                   'Go back if letter before colon is "H"
80 FOR II = I-2 TO I+5
90 TIME$=TIME$+CHR$(PEEK(II))
100 NEXT
110 PRINT "LOWER ADDRESS="II-8" UPPER ADDRESS="II-1" TIME= "T$
120 END
```

Listing A.

```
100 '******READTIME.BAS*******
110 'READ THE TIME IN THE CLOCK BY CHARLES HORN
120 FOR I = 4903 TO 4910                    ' Memory locations from FINDTIME.BAS
130 TIME$=TIME$+CHR$(PEEK(I))
140 NEXT
```

Listing B.

ed to Perks). My thanks to their authors for resolving these issues.

I would also like to point out that while the Perks calculator was described in the article as a floating-point calculator, it actually uses decimal and not binary floating-point arithmetic. This precludes the rounding errors which are so common with other floating-point calculators and which make them unsuitable for financial calculations.

Finally, the reviewer commented that Perks' polish "wears thin" in the import mode. While the item to which the author was referring will be addressed in version 2.X, for clarification it should be noted that what the author was talking about was the ability to read a disk file into the editor buffer, and *not* the "import" function as defined in Perks' documentation. The latter, which is Perks' ability to copy the underlying screen into the editor, is one of Perks' strongest and most elegant features and, unfortunately, was not discussed in the article at all.

As part of our customer support, we have provided *free* updates (not even charging for postage) to all of our existing customers with versions earlier than 1.13, upon return of their original disk to us. This practice will continue for Perks versions 1.X, although there will be a nominal charge for upgrading to version 2.X, which we expect will be available by the time this is published (March-April). Version 2.X will include entire new modules, enhancements to existing modules, and a variable-size (2K to 64K) notepad, while still remaining small in size (in the 64K to 80K range, plus the notepad buffer) and still working with both Z-DOS and MS-DOS. We will notify our customers of the availability of 2.X when it is appropriate to do so.

Thank you for your support of the Heath/Zenith community.

Barry A. Watzman
Benton Harbor, MI

Omissions and inconsistencies on Genie

The "Desktop Utilities" article has a few omissions and inconsistencies. First your readers should be aware of your six-month lead time. A period of six months between the initial review of a product and the publication of a review is irresponsible, particularly in an industry as volatile as the computer industry. Many changes have occurred in Genie and its competitors in that period of time, and since you were not willing to let the publishers of the various programs see the article or send in comments prior to publication of the review, you perhaps should have made more of an effort to keep the review up to date. Be that as it may, here are some of the errors I found:

1. Genie now uses only 130K when fully configured, if one has 64K video RAM parts.

2. A fully configured Genie consists of: key macros, DOS services, both calculators, an 8K notepad, the cut and paste module, a "Rolodex", an alarm clock, an ASCII table, a typewriter, a screen saver, and a 2K MS-DOS command stack.

3. The Genie calculator can not now, and never could, perform conversion to or from base 2.

4. The Genie note pad now has a print feature to print the contents of the note pad directly.

5. The Genie ASCII table has 256 characters on the IBM PC version, and I might point out that values above 127 are not considered part of the ASCII character set, and therefore should not be included in an ASCII table. On the PC, values 128-255 do have displayable meaning, and they are included on the PC version of Genie.

6. And most important: I think you glossed over the major difference between Genie and the other members of the desktop utility world; and that is: Genie has *many* more features. The article seemed to spend a lot of time detailing the business features such as the note pad and calendar, but skimmed over the features, such as the DOS services and key macros, that programmers and engineers might find interesting.

Michael L. Spilo
Advanced Software Technologies
New York, NY

The lead time between the writing of an

article and its publication is something a bimonthly magazine such as Sextant *must always take into consideration, and we often make last-minute changes to product reviews in light of price changes and other modifications. However, our review of the desktop utilities put us in a different situation. Here there were five products, and not just one; and here we knew that no matter how many revisions we made, the products would probably continue to change even while the magazine was at the printer. Therefore, we set a cutoff point, last fall, after which we did not change the substance of the review. We took great pains to point out in the article that it was not current. The closing lines of the review noted this, and we included a box devoted to the subject of the evolution of these products.*

We invited the authors of the programs to respond with letters to the Editor; we're glad that two of them have taken the opportunity to do so.

Save WordStar ruler lines with file

Reference: Letter from James G. Barr, issue #19 (November-December), page 4.

I have been intending to write this letter since I got the referenced issue of *Sextant*, November 8, 1985. It may be that this subject, which was on Mr. Barr's wish list as a second item, has already been covered by other readers. In any case, here is the solution:

1. Type in the desired ruler line, using dashes and exclamation marks where desired for tabs.
2. With the cursor in column 1 of the ruler line and in the insert mode, type two periods, type ^P, and then press RETURN or ENTER.
3. Place the cursor on the same line as the ruler and execute a ^OF (not ^O^F) to make that ruler line active for that file. (The ^OF command must be executed every time there is a change to a different ruler line, because WordStar will retain a ruler line until changed or until there is a reboot.)

This procedure provides a non-printing ruler line that is saved with the file. It is suggested that the normal ruler line be turned off with ^OT when creating and using different ruler lines because it serves no purpose and uses up a line on the screen.

The above information is available in several publications, but the ultimate, veritable "gold mine" of WordStar tips and technical information is available from:

"Under Ground WordStar"
Hard/Soft Press
Box 1277-A
Riverdale, NY 10471

Back in August of last year I received the manual (the treasure chest!) and a "treasure disk" from the Hard/Soft Press. It cost me $21.95 at that time (the price may have changed), and I would have gladly paid several times that amount to get it. It contains all of the patches to WordStar, version 3.3, known to be in existence at the time, and puts it all into a small soft-cover manual. Besides the patches, the manual includes important addresses for functions that can be changed using the MS-DOS DEBUG utility, and many tips and technical information, including automatic back-up saves when using WordStar with a RAM disk. As examples of what is in the manual: how many users know that after typing in the name of the file to be printed you can use the ESCape key to execute all the defaults instead of stepping through them with RETURN? A simple tip: when changing logged drives, it is not necessary to place a colon after the new drive letter. (WordStar is smart enough to know this.) And it goes on and on.

I apologize to Mr. Barr for taking so long with this information, but I hope that the additional information about "Under Ground WordStar" compensates for the delay. By the way, on the cover of the manual the authors are identified as "Ward Star" and "Mel Murch"! The authors' names may be tongue-in-cheek, but the information sure isn't!

Paul W. Stuehn
Canton, MI

Please send letters to: Editor, Sextant Magazine, 716 E Street, S.E., Washington, DC 20003. Letters will be edited according to style and space requirements. Due to the volume of mail received, we are unable to print or respond to every letter we receive.

Ultimate Communications Software

HyperACCESS™

A Dramatic Step Beyond ACCESS™†

HyperACCESS gives you tremendous power—power that lets you communicate without getting technical—power that lets you glide through applications other programs can't touch.

A New Standard for Ease of Use

A startup program sets up everything for you and a fascinating on-screen tutorial lets you try out program features in simulated use of Compuserve, MCI Mail, Easylink, and others. HyperACCESS menus are instinctively easy to use, so you can focus on your work, instead of on how to run the program.

Effortless Modem Support

Instantly call any remote system by selecting it from a menu that lets you list up to 127 systems and already lists Compuserve, The Source, MCI Mail, Easylink, and other systems (subscriptions not included). The first time you call these systems, your password and user ID are memorized, and they are entered for you automatically, thereafter.

Remarkably Flexible

HyperACCESS comes ready-to-use for the more popular applications and it lets you set and store all the details you need, to connect with main frames, minis, micros, or RS232-compatible devices that defy other programs.

Terminal Emulation

HyperACCESS emulates the most popular terminals, including DEC VT-52, VT-100, TeleVideo 900 series, IBM 3101, and H-19, to let you run programs on remote computers designed for use from terminals.

Use Your Computer's Power

From within HyperACCESS, you can use your computer's DOS commands (DIR, COPY, etc.) or run your other programs. You can even define DOS macros, so you can execute a series of DOS commands and/or programs by pressing a single key.

Use Xmodem, Kermit, or Text Transfer Methods

. . . to reliably send/receive files of any kind from/to any disk drive or directory in your computer, even through noisy telephone lines.

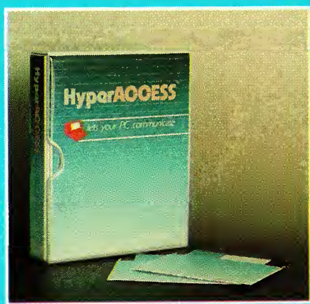
Remote Use of Your Computer

Press one key and, even when you're absent, operators of remote terminals or computers can access your computer by entering passwords you define. You can set whether each password allows a user to read or type messages, transfer files, run a given program, or use your operating system and all your programs.*

Automate Your Communications

. . . so you can perform, with a keystroke, steps that normally would take many keystrokes. HyperACCESS has the clearest, most comprehensive script language available, to make automating communications easy for beginners and exhilarating for experts. You can automate any part of a call, make unattended calls, or create custom functions, prompts, and menus.

For astonishing power and versatility, get HyperACCESS today!



For PC-compatibles and Z-100 computers with DOS and any modem, including 2400 baud and USR S100 modems. You can get HyperACCESS for $149 from computer dealers or from Hilgraeve Inc. (please specify computer and add $5 shipping).

Hilgraeve Inc.
P.O. Box 941
Monroe, MI 48161
(313) 243-0576

Z100 version is fully compatible with Gemini and Easy PC boards.

Circle #203 on Reader Service Card

† Hilgraeve and its dealers will continue to sell ACCESS, as well.
HyperACCESS and ACCESS are trademarks of Hilgraeve Inc.

* Certain programs cannot be operated remotely.

Free Nanoseconds Highlight CHUGCON 85

Commodore Grace Hopper put technology into perspective as users relaxed after a long day of seminars and bargain-hunting.

Victoria Saxon

Commodore Grace Hopper was thoughtful enough to bring so many nanoseconds to the CHUGCON 85 banquet that everyone who attended was able to bring one home. My particular nanosecond (which was green, by the way) fit very nicely in the bottom of my handbag, where it stayed all during the subway ride home later that evening.

I don't know if you can really understand this, but there's something pretty neat about knowing that you're probably the only person on a whole subway who is actually carrying a nanosecond. I figured that most of those people on that subway didn't even know what a nanosecond looked like. In fact, they probably didn't even know that if the first guy who ever made a clock had made it run counter-clockwise instead of clockwise, we still would've been able to tell time.

Actually, in order to be fair about this, I have to admit I had certain advantages on that subway. You see, I had just spent the day at CHUGCON 85, the 1985 International Conference of the Capital Heath Users' Group. Now you may not think that this is any big deal, but let me tell you something: spending a day at CHUGCON means about 14 hours or so spent going to tutorials, talking to vendors, having conversations with other computer enthusiasts, and trying desperately not to spend your entire month's paycheck.

So what does this have to do with my being the smartest person on the subway? Let's put it this way: Some people get the meaning of life by reading James Joyce; I got the meaning of CHUGCON by listening to Commodore Hopper's speech at the CHUGCON banquet. After all, what can be more thrilling than to learn all about backwards clocks, nanoseconds, and the very future of the microcomputer industry?

Of backwards clocks and nanoseconds

Commodore Hopper was a pretty special banquet speaker. She's been working with computers for more than 40 years—first on active duty in the Navy during World War II, and then as a civilian and in the Reserves. In 1967, she was recalled to active duty and has been working for the Navy ever since. She currently holds the title of Special Advisor to the Commander of the U.S. Navy Data Automation Command.

However, her accomplishments are almost more impressive than her titles. This is the woman who wrote the very first compiler. She was also extremely influential in the development of the COBOL programming language. It's enough to make you want to listen to her when she talks about computers.

> *When one ox wasn't big enough to move the tree stump, they got two of them.*

According to the commodore, the "most horrible phrase" you can ever hear in the computer industry is "but we've always done it that way."

For this reason, she has a clock in her office that runs counter-clockwise. It doesn't take very long, she claims, for people to figure out that it really doesn't matter how you've "always" read the clock. And it doesn't matter which way those hands on the clock turn. You just learn to read the time backwards (or "forwards" if you really get into it).

She claims to have another clock whose hand stands still while the digits rotate around in the background. I don't dare ask what happens to the minute hand—the commodore threatened to haunt any one of us who over the next twelve months dared to hint at saying "but we've always done it that way."

One of the first computers the commodore came across was the Mark I. The Mark I was donated to Harvard University by IBM in 1944. It was the world's first large-scale digital computer, and she was its third programmer.

The Mark I measured 51 feet long by 8 feet high, and according to Commodore Hopper, it was capable of "calculating three additions every single second." By 1951, computers had increased their speed to one calculation every 282 microseconds; and by 1964, one every 300 nanoseconds—and the size of the computers had decreased significantly.

What next? she asks—perhaps one calculation every 300 picoseconds? Believing this is entirely possible, she emphasized "we need it right now."

Like many people, she had been a little puzzled by the term "nanosecond" when it was first used in her presence. Reacting to complaints of wasting nanoseconds in her programs, she asked stubbornly that someone show her exactly what a billionth of a second was. It certainly couldn't be seen on a clock.

Eventually, in response to her requests, she was presented with the same thing which she gave to all of us at the banquet: a piece of wire that measured just over eleven inches, and which represented how far light could travel during one billionth of a second. Now that's food for thought the next time you're talking about wasted nanoseconds in a computer program. Just think, one little nanosecond is almost a whole foot long.

As Commodore Hopper's career progressed, she came in contact with more and more computers, and she soon learned that essentially there is not enough time for anyone to keep up with all the technical advancements which are constantly being made. As a result, people often begin to feel threatened by computers.

She quoted one hysterical captain who wrote to his admiral, "Do something—they're all over my ship." What he meant, explained Hopper, was that his ship had been "invaded by microcomputers." What a frightful experience.

The solution? Learn to listen to our "juniors." In other words, don't be scared of all these first-graders who are already becoming computer whizzes. Instead, learn to communicate with them and provide them with leadership and guidance. They don't know everything, but bright, young, innovative people are often able to give good, innovative suggestions.

The speech was filled with anecdotes and good humor, and I doubt that anyone escaped from that banquet without being influenced by one of her lessons.

The "I-remember-when" stories were fun, educational, and numerous. It is important to realize, though, that this is not a woman who has chosen to live entirely by the rules of the past.

In fact, if there was one point which stood out from the rest of the commodore's advice-filled speech, it was the statement, "We've got to move into the future." Whether this involves realizing that "backwards" clocks can really function or realizing that we do live in an age in which nanoseconds make a difference, the advice remains the same.

In making decisions, it is just as important to realize "the cost of not doing something" as it is to realize "the cost of doing it." A failure to "implement the standards" can be more harmful than good in the long run when replacement and improvement costs begin to be calculated.

But what, exactly, does that mean?

In short, we must learn to curb the excitement which encourages us, for example, to buy "hardware for its blinking lights." Instead, use some forethought, and find out what will be most useful in the long run.

Back in the olden days, lectured Commodore Hopper, when farmers had to move big stumps, they used to get an ox to pull it out of the ground. When one ox wasn't big enough to move the tree stump, they didn't try to grow a bigger ox, they got two of them.

Finally, while stressing the importance of communication, the commodore informed us that it is not just the information which computers give that is important. There is at least one aspect of the computer industry which must not be forgotten: communication among human beings.

In addition to the person who feeds information to the computer, there is the person to whom the information is given. The final process of a computer analysis is the human being who actually organizes, turns the information into knowledge, and presents that knowledge to someone else in "plain English." Without that final, human ability, an important communications link may be lost.

At the end of her speech, Commodore Hopper received an immediate and unanimous standing ovation. Her words had been inspiring and refreshing. This had not been a typical lesson of the old to the young, or of a computer scientist to a literature major. It was, instead, a plea for communication.

Her speech combined experience and a sincere desire to look forward. Education and leadership, a serious consideration of the future, the forethought to implement changes (or, in some cases, to resist the temptation to do so), and communication—all of these are essential factors in making the computer industry as progressive as possible.

Now let me show you how this clarified CHUGCON; or, more appropriately perhaps, how CHUGCON lived up to the standards.

Education and the Air Force

CHUG's president, John Roach, described CHUGCON 85 as "a very good excuse to bring together an awful lot of the Heath/Zenith community all in one place at one time, so that we could simply enjoy being together. You get to see a lot of things you don't normally see in a year, and you get to hear a lot of things in the tutorials."

> ***What happens if you find that you can get this really great deal on five oxen for the price of three?***

One of the major themes of this conference was education. Three instructors from the U.S. Air Force Academy—Major Mark Tollefson, Major Lee Schrock, and Captain Louise Blanchette—gave tutorials dealing with the subject of computers and education.

Unfortunately, I missed one of these tutorials, "Computerized Test Delivery Network for $5 a Node," given by Schrock and Tollefson. This testing system is apparently capable of handling up to 6,000 students. What's so great about that? The system can give each student a unique test equal in difficulty to those of his peers. No more cheat sheets for the people taking later classes.

You could tell that Major Tollefson was used to teaching. In his talk "Human Engineering of Computer Education," he emphasized that programs for students must be, among other things, pleasant to use, quick, interesting, fun, and bomb-proof. ("...[P]resume your students are going to lean against the keyboard.")

The HELP key, he claimed, is of great importance. If students ask for help, they shouldn't be given five pages of boring documentation, or they'll never go back to the key for help again. Instead, when they press the HELP key, they should find a fun and educational story—perhaps even the fictional character of "Captain Kirk explaining a problem to Scotty." This may make your self-help program not only more entertaining, but also more inviting for your students to use.

I was particularly impressed by Captain Louise Blanchette's demonstration and talk on her "Calculus Tutorial" program. Captain Blanchette wrote the program for her students who had either reached an advanced level of calculus and needed to review some of the earlier stages, or who had just begun calculus and needed some extra help and practice in learning.

She stressed that the program cannot take the place of a teacher, so students should use it only after having become at least vaguely familiar with the concepts in class. Nonetheless, she emphasized that the program may be used to help students work to help themselves.

I tried the program and was very pleasantly surprised. It's been three years since I took calculus, so with this program I started at the beginning level. Since I was sure I didn't remember anything, I was not at all surprised when I gave the wrong answer to the first question.

But then something amazing began to happen. After the program explained to me how to do the problem correctly, I gave the right answer to the next question! Soon I moved on to other questions, and before I knew it, I had progressed to the "hard" level!

In no more than ten minutes, I was (much to my dismay) reminded of what a calculus derivative was; and, believe it or not, I actually was capable of doing some problems. (There was, of course, all the rest of calculus sitting there in the program, waiting for me to review it, but I never got that far.)

The program is not quite ready for general distribution, but Captain Blanchette said that a beta-test version of the object code would be available for the Z100 and the Z150 by the end of January 1986. There are two disks—one for differentiation and one for integration. They're free, too (which, of course, makes them even more inviting).

The future...?

One hot topic at CHUGCON 85, as elsewhere, was getting the H/Z100 to run software designed for the IBM Personal Computer. The two commercial contenders offering boards in this area are UCI Corporation and Gemini Technology (formerly D.E.L. Professional Systems). They're both offering PC-emulator boards, and the two products came out on the market almost at the same time. Both companies had exhibits at CHUGCON and both gave tutorials on their boards.

The two boards reflect two different approaches. UCI's Easy PC is a three-

board package that takes more of a "hardware" approach, and takes up one of the Z100's S100 expansion slots. The Gemini board, on the other hand, takes more of a "software" approach. It is a two-microprocessor board which plugs into the socket of the Z100's 8088 central processor chip. Your original 8088 takes a place on the Gemini board.

Unfortunately, I wasn't able to get to the UCI demonstration, but I did attend the Gemini talk. When I was there, the downstairs theatre at the Hyatt began to seem more like the briefing room for futuristic star fighters than a conference for a group of computer enthusiasts. (Of course, some would like to argue that there's not that much difference.)

Needless to say, the room was packed for their presentation. On the desk in front of every chair was placed a slick-looking color booklet entitled "Gemini Enhanced: The Story of Gemini Technology Inc." with some Star Wars–like pictures of the Gemini board on the cover.

Gemini's president, Ed Dolejsi, was there along with the director of research and development, Peter Van Baarsen. They started their presentation with a film that stressed the importance of enhancing a computer like the Z100 for IBM-PC compatibility. The film claimed that the Gemini project might be regarded as "the birth of an industry."

Afterwards, the audience had the opportunity to listen to Van Baarsen explain, simply (thank goodness!), how the board works. Basically, he claimed, the board is "really a glorified 8088 processor."

Relaying, in his own words, a "simple idea," Van Baarsen described the Gemini: Since the Z100 has no IBM hardware, something must obviously be done when a program makes a call to an IBM hardware feature.

Of the Gemini board's two microprocessors, one handles normal operation until there's a call for an IBM feature. When that call is intercepted, the second processor takes over. This processor figures out what has to be done and then translates the concept to the Z100. The second processor, Van Baarsen said, "basically tells the Z100 in the language it will understand to go to the floppy drive, to put this on the screen, to read something from the keyboard, and then returns it to the processor that's running the IBM software."

The Gemini works. There are, however, a few admitted drawbacks. First, the Gemini can be as much as 10-15% slower than the IBM in some cases—especially in the text mode. (The graphics mode can be faster.) Second, operating systems for both the '100 and the IBM cannot reside in memory at the same time; so, it is necessary to do a hardware reset to switch between IBM and Z100 modes.

Van Baarsen was also asked to compare the Gemini board to UCI Corporation's Easy PC. In response, Van Baarsen commented that the Easy PC is "the other approach to the problem." He said that the Easy PC is "more or less a computer within a computer," whereas the Gemini relies more on an emulator program stored in its read-only memory (ROM).

Van Baarsen claimed that the advantages the Gemini has over the Easy PC are that it can take an 8087 math co-processor, that it does not use up any of the Z100's S100 expansion slots, and that it has direct access to the serial input/output (I/O) ports rather than having to depend on the operating system.

So what does Van Baarsen think are the *disadvantages* of having a Gemini instead of the Easy PC?

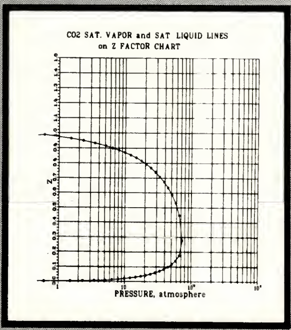
"As far as I know, our board is limited in some ways as far as video, and that's because of limitations of the Z100."

Forethought (or, buying a second ox)

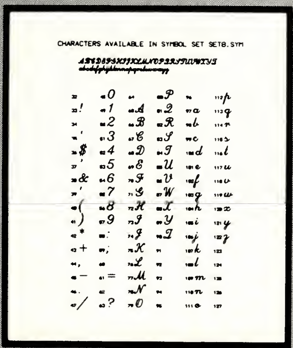
I don't know if Commodore Hopper's farmers of the olden days were given the opportunity to attend oxen conferences. If so, I bet they had some of the same problems that HUGgies have. I mean, just imagine for a second that you're a farmer in the olden days, and you know you have to go to this oxen conference to buy a second ox to get that stupid stump out of your fields.

GOLDEN SOFTWARE HAS THE GRAPHICS YOU NEED

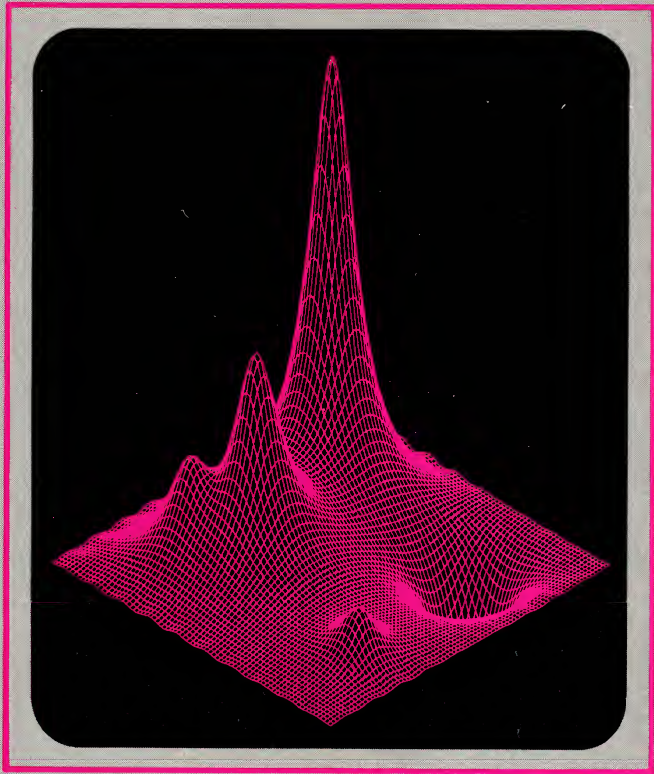
Send for DEMO DISK Only $10

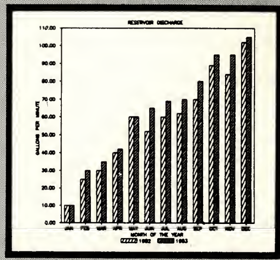


Scientific Plotting



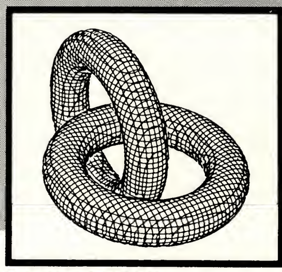
Symbol Sets



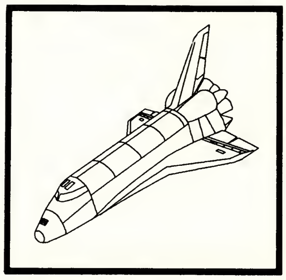


Business Graphics

Contouring



Program Output



Digitizing

GOLDEN GRAPHICS SYSTEM plots two dimensional XY data, three dimensional XYZ data, or can process standard plotter instructions on a dot matrix printer. The **GOLDEN GRAPHICS SYSTEM** includes the following features:

CONTOURING

Up to 3,000 random data points (irregularly spaced) are read from the keyboard or from a data file to produce a contour map with in-line contour labels. All options such as frequency of labeled lines, map size, posting, contour limits, interpolation method (Kriging or Inverse Distance Squared) and titles are easily specified from a menu.

3-D SURFACE PLOTS

You may also interactively create perspective block diagrams. Up to 3,000 random data points (irregularly spaced) are read from the keyboard or from a data file to produce a three-dimensional surface. All options such as viewing angle, hidden line removal, titles and plot size are easily specified from a menu. All plots in this advertisement were created with the **GOLDEN GRAPHICS SYSTEM.**

GRAPHS

Interactively create XY graphs with linear-linear, log-linear, linear-log, or log-log axes. An infinite number of data pairs may be read from the keyboard or from a data file to produce a presentation quality graph. Bar charts, pie charts, and line graphs may also be created.

PLOTTER COMMANDS FOR PRINTERS

Convert your dot matrix printer into a plotter. Easy to understand plotter instructions that have been placed in a file may be processed for output to a printer or plotter. Seventeen symbol sets are available for plot labeling and sign making. Character strings may be created from any symbol set, rotated to any angle, and scaled to any size. Plots can be created from any language.

Requirements

The only requirements are: H/Z100, H/Z150, IBM PC or workalike, DOS (any version), 128K, and a dot matrix printer or plotter. All plots are created directly on the printer or plotter to achieve the maximum resolution possible. This is not another print screen program; printer drivers have been developed for over thirty different printers for maximum resolution (120 x 216 dots per inch on the Epson MX-80).

GOLDEN GRAPHICS SYSTEM $299

Printer/Plotter Demo Disk $10
(include printer/plotter model)

VISA, MasterCard, American Express and purchase orders accepted

To order or obtain further information use coupon or call:

1-800-972-1021
(in Colorado: 279-1021)

—Money back guarantee.—

Mail to: **Golden Software, P.O. Box 281**
Golden, CO 80402

☐ YES! Send your $10 demo disk.
☐ Please send me your free brochure.
☐ H/Z100 ☐ H/Z150-IBM

Name: ____
Affiliation: ____
Address: ____
City: ____ St: ____ Zip: ____
Printer or Plotter Model: ____
☐ VISA ☐ MC ☐ AM EXP ☐ PAYMENT ENCLOSED
Acct. No.: ____
Exp. Date: ____
Signature: ____

GOLDEN SOFTWARE, P.O. Box 281, Golden, CO 80402 - (303) 279-1021

What happens if, when you arrive at the convention, you find that you can get this really great deal on five oxen for the price of three? Or what if you know you're supposed to bring oatmeal back for your starving children, but this one guy has this fantastic deal on ox shoes—and you know that if your oxen had better shoes, then they could move more stumps, and you could grow more food, and your children wouldn't starve?

My guess is that life before computers really wasn't any easier than now.

At CHUGCON, the temptations were not ox shoes, but they were almost as enticing. The Heath/Zenith Computers & Electronics Centers' "booth" covered the largest single section of the floor area in the exhibit hall, and they made good use of the extra space.

I think they brought the whole store with them and began immediately dealing with slashed prices. By Sunday afternoon, the portable ZP-150 was selling for $749 (list price was once $1,995). They were also selling Gemini Emulator boards for $499 with software, and the single-drive portable ZFA-161-21 for $1,299.

Over in a different section of the vendor area, William Perry of Trionyx Electronics was showing off his new 16-bit processor board for the H8. "It's been a real challenge getting this guy, but we finally got it," he said.

The board, the 16B-H8, takes either an 8088 central processor (also used in the Z100) or an 8086. It also has an empty socket for an 8087 math co-processor ("which everybody seems to want"). The 16B-H8 presently supports operation under CP/M-86; Trionyx is "ultimately" working towards PC-DOS.

The board can talk to the H8's 8-bit boards, such as Trionyx's CH8 disk-controller board and Z80 processor board. And it can support the IBM keyboard and speaker.

A second new board from Trionyx is a memory card plus memory management. "Byte-swapping" operation allows you to use both the 8-bit and the 16-bit processors without any need for reconfiguring the board.

Perry happily commented that the logic to do all the byte swapping is contained in programmable-array logic (PAL) chips—which apparently were not easy to get.

Later, I passed a crowded booth, and when I peered through the mobs to see the cause of all the excitement, I saw that new H89 kits were being sold for $150. The vendor was Al Davis Enterprises. Davis said he was selling circuit boards for all the latest computers for $35; other items, such as H/Z89 terminal logic boards, were going for $5 each.

Most of the vendors had a few special deals for the show, and some were even stocked with little gifts to be remembered by. Graymatter Applications Software was giving out free pens; D.E.L. Professional Systems/Gemini Technologies, Inc. had Gemini cuff link/tie tacks; and Weber & Sons was handing out samples of their "Kangaroo" disk-jacket "pocket." The adhesive pockets stick onto disk jackets, and allow harried computer users to write labels on replaceable pocket inserts.

Mark Wolinsky of Floppy Disk Services, Inc. reported big sales of their high-capacity 5¼″ drive called the "Hot-Setup." It's like the 1.6-megabyte drive that the IBM AT uses. The drive can be used with the Z100 and the H89 with a special cable. It can also be used with the Z150 and '160, but in these cases the addition of a disk controller board and supporting software is necessary.

Essentially, says Wolinsky, "It's a 5¼″ drive that looks like a double-sided 8″ drive electrically, so it holds one and a quarter megabytes per floppy." With one such system on display at their booth, Floppy Disk attracted enough attention to sell every piece they had, and they even took some orders home.

Communication

Communication really seemed to be the main theme at CHUGCON, and it came in several different forms. Joseph

Schulte, president of Veritechnology Electronics Corporation gave a talk entitled, "Heath Electronics Centers—Past, Present & Future." Among the more exciting announcements Schulte made was that their stores would be carrying Apple (yes, Apple) products in the future.

Schulte also showed us a picture of Edward Bayard Heath, who founded the Heath Company in 1926. Even Schulte seemed a little awed by the fact that Ed Heath started his company by "designing, manufacturing, and selling airplane kits"!

Schulte also explained the difference between the Heathkit catalogue and the Veritechnology catalogue: Heathkit has an order form in the middle and a picture of president Bill Johnson on the back cover, whereas the Veritechnology catalogue has a picture of Joe Schulte on the back cover, and has a slightly different set of phone numbers in the middle. (Don't laugh—these kinds of things can be very confusing if you don't know what's going on.)

Zenith Data Systems sent Mark Foster, their chief systems architect, to tell us about their new machines—the Z138, '148, '158, '171, and '200—all IBM compatibles.

Now, to tell you the truth, I half expected to be completely confused by a speech from somebody called "chief systems architect," but Foster proved to be someone who, in my book, is the ideal "communicator." Without going into Warp-factor-five computer language, he assumed a certain minimal amount of knowledge on the audience's part, and then proceeded to tell us all the fun stuff.

Beginning with the '148, he described it as ZDS's version of the PCjr.—only unlike the other guys, ZDS didn't "neuter" their machine. (That is, it's not stuck with just one disk drive.) He described all the great features of the '148 (good video output with an inexpensive monitor, expansion options, the accelerator switch which allows the '148 to speed up from 4.77 megahertz to 8 MHz, etc.). Finally, he implied that the best part about the '148 is that you can get a pretty decent computer without paying a whole lot of money. (I told you I liked this guy.)

The '138, he said, is basically the same as the '148 except that it's transportable, and it has a built-in monitor. It also has a full-size, unaltered keyboard—which is unusual among transportable computers. The '171 was described as a "rather odd-looking machine, but it's very small ... a real portable."

Rather cautiously, Foster told us that the "Z158 is the successor to the Z150. Yes, that means we're not selling the Z150 anymore." However, Foster explained, ZDS will still be supporting the '150. The same ROM is used with the '158—so the '150 will benefit from future upgrades. Even the version numbers are the same for the '150 as for the '158.

With a number of improvements over the '150, the '158 has been designated the "new standard machine." Like the '150, it offers fast, flicker-free video. And it runs at a faster clock speed than the '150. (You can run at both the '150's 4.77 megahertz and at 8 MHz.) Also, the '158 can take the newer 256-kilobit memory chips, so you can get a lot more memory on a board.

For program developers, the '158 has another nice addition: a full-blown debugger in ROM (to let you execute a program step by step, stop it where needed, do changes, and go back to where you left off).

Foster saved the best for last: "the flagship—the Z200: our equivalent of the U.S.S. Enterprise—and it's designed absolutely from day one for speed and the best and largest expansion capability we could fit into it." Compatible with the IBM PC AT, the '200's memory can be expanded up to 15 megabytes. Don't get too excited, though. As Foster quipped, there will probably be some program coming out within the next year which requires that much memory.

Just think, one little nanosecond is almost a whole foot long.

Finally, if we're going to talk about good communication, Susan Hayes of The Software Toolworks deserves a gold star. She gave a lecture entitled "MS-DOS for Beginners."

Before she started, she politely informed those people who had listened to her lecture before that they should feel free to leave if they wanted to, since she would be repeating the same beginners' lesson she had given before. Thus, leaving all potential for awkwardness behind, she wrote the words "software" and "hardware" on the blackboard and proceeded to define those two words before any of us could get too embarrassed.

Reassuring us that she had done everything that could possibly be done wrong on a computer, she made everyone feel comfortable asking their deepest, darkest, most ludicrous questions. Now don't you wish that someone like Susan Hayes was around when you got your first computer?

The next generation?

All in all, I have to say CHUGCON was not only educational, but fun and exciting, too. Now I know all about nanoseconds, backwards clocks, and lots of other neat things.

My only real misgivings arose from the fact that I left the conference thinking that maybe I wouldn't be able to fulfill my obligation to communicate with the next generation properly.

I don't even feel that old, but once during the show, I walked out of the demonstration area to find a small child at the pre-toddler stage crawling in hot pursuit of a HERO Jr. The robot, who seemed to be travelling just fast enough to stay out of reach of the unsuspecting child, was crooning a somewhat off-key version of "Daisy, Daisy, give me your an-swer true. I'm half crazy o-ver the love of you."

I guess, as Commodore Hopper suggests, we should be ready for all kinds of changes. So don't be surprised if next year's conference is attended by an up-and-coming generation of two-year-olds going to lectures given by robots.

Or vice-versa.

Sources Mentioned

Al Davis Enterprises
187 Lake Chapin Road
Berrien Springs, MI 49103
616/471-1792

Captain Louise Blanchette
USAF Academy
Office DFMS
Colorado Springs, CO 80840
303/472-2070

Floppy Disk Services, Inc.
39 Everett Drive, Building D
Lawrenceville, NJ 08648
609/799-4440 (information or technical support); 800/223-0306 (orders)

Gemini Technology, Inc.
#11-1151 Horseshoe Way
Richmond, B.C.
Canada V7A 4S5
604/274-1283 (information and technical support); 800/663-8331 (orders)

The Software Toolworks
14478 Glorietta Drive
Sherman Oaks, CA 91423
818/986-4885
Orders: 800/223-8665; 800/228-8665 in California

Trionyx Electronics, Inc.
P.O. Box 5131
Santa Ana, CA 92704
714/830-2092

UCI Corporation
948 Cherry Street
Kent, OH 44240
216/673-5155, 800/UCI-COMP

Weber & Sons, Inc.
P. O. Box 1027
Adelphia, NJ 07710
800/225-0044; 201/431-1128 in New Jersey △

BASIC Questions and Answers

Edward A. Byrnes

Creating a stationary object with moving parts

Creating a travelling object with moving parts

Moving an image with the right and left arrow keys

Moving an image right, left, up, and down

Using the keypad for up, down, right, left, and diagonal travel

Sending shells from moving objects

Experimenters' notes

In *Sextant* #18, you gave a number of pointers on creating graphic pictures. Could you do the same for computer animation? I know how to get things on and off the screen; I'd like to be able to make them move.

That one question actually breaks into two smaller questions. First, you have to make an image vary while it's sitting in one position on the screen. (A man waving his arms comes to mind.) Then you have to move that moving image around the screen.

I'll take you through the steps required to create an animated sequence. Listing 1 provides a simple animation program; Listings 2 through 6 add increasing levels of complexity. The lines in each listing should be added to the previous one, and in some cases a few lines should be deleted, as indicated in the caption.

(The numbers in the REMark at the end of each program line indicate which listings that line can be found in. So if you wanted to type in the fourth program without first typing in Listings 1 through 3, for example, you could simply type in every line, of all the listings, that had a 4 among the numbers in its REMark.)

(Listings 1-6 were written to run on the Z100. With a few changes, all the programs listed here will run on the Z150 under GW BASIC. The program changes for the Z150 are given in Table 1.)

Creating a stationary object with moving parts

If you can make objects appear and disappear on the monitor, then you are only a few steps away from making it seem as if these objects are moving.

Creating animated sequences on the computer is similar to creating animated cartoons. If you want to make a character appear to run in a cartoon, you will draw a series of pictures of the character in different running positions. These drawings will then replace each other in sequence and appear in different places on the drawing medium. If the drawings replace each other fast enough, the illusion of movement is created.

We can create the same type of illusion on the computer screen with Z-BASIC's GET and PUT commands. (For more on GET and PUT, you might want to look at "Standard Operating Procedure" in *Sextant* #15, March-April 1985. See "Z-BASIC Graphics: Using GET and PUT," by Phil Winninghoff.)

For a simple example, look at MOTION1.BAS, in Listing 1. This is a Z-BASIC program to create a stationary "helicopter" with a moving rotor blade.

The helicopter is created in lines 1040-1070. The CIRCLE command in line 1040 gives us an oval shape for the body of the helicopter—center at (320,111), radius of 12, drawn in the color yellow (6), going

```
1000 REM ***** PROGRAM 1 FOR THE Z-100
1010 REM *****
1020 REM *****
1030 COLOR 6,0:CLS                                          :REM 1,2,3,4,5,6
1040 CIRCLE(320,111),12,6,0,2*3.14159,.2                    :REM 1,2,3,4,5,6
1050 PAINT(320,111),4,6                                     :REM 1,2,3,4,5,6
1060 LINE(320,108)-(320,105):LINE(319,108)-(319,105)        :REM 1,2,3,4,5,6
1070 LINE(318,105)-(322,105)                                :REM 1,2,3,4,5,6
1080 DIM A#(100):GET(305,105)-(335,118),A#                  :REM 1,2,3,4,5,6
1090 LINE(314,105)-(326,105)                                :REM 1,2,3,4,5,6
1100 DIM B#(100):GET(305,105)-(335,118),B#                  :REM 1,2,3,4,5,6
1110 LINE(310,105)-(330,105)                                :REM 1,2,3,4,5,6
1120 DIM C#(100):GET(305,105)-(335,118),C#                  :REM 1,2,3,4,5,6
1130 PUT(305,105),C#:B=305:E=105                            :REM 1,2,3,4,5,6
1170 REM ***** START PUT SEQUENCE                           :REM 1,2,3,4,5,6
1180 PUT(305,105),A#:GOSUB 1670                             :REM 1
1190 PUT(305,105),A#                                        :REM 1
1200 PUT(305,105),B#:GOSUB 1670                             :REM 1
1210 PUT(305,105),B#                                        :REM 1
1220 PUT(305,105),C#:GOSUB 1670                             :REM 1
1230 PUT(305,105),C#                                        :REM 1
1430 ZZ$=INKEY$:IF ZZ$="Z" OR ZZ$="z" THEN END              :REM 1,2,3,4,5,6
1660 GOTO 1170                                              :REM 1,2,3,4,5,6
1670 FOR C=1 TO 75:NEXT:RETURN                              :REM 1,2,3,4,5,6
```

Listing 1. MOTION1.BAS. This Z-BASIC program will display a stationary helicopter with a moving rotor blade. Lines in this listing will serve as the basis for more complex animation. (See Listings 2-6.) REMarks indicate which lines are used in programs MOTION1 through MOTION6. As given in Listings 1-6, the programs can be run on the H/Z100 under Z-DOS/MS-DOS. See Listing 7 for changes to be made to run them on the H/Z150 under PC-DOS/MS-DOS.

the full circle (starting at 0 and ending at 2 pi), and with an aspect ratio of .2. This is then filled with the PAINT command, in the color red (4), with a yellow (6) border.

The first two LINE commands, in line 1060, place two vertical lines side by side, centered at the top of the oval; this creates an "upright" to hold the rotor blade. The third LINE command provides us with a short rotor blade at right angles to the upright (line 1070).

For the coordinates in the GET command in line 1080, we can use the boundaries of the screen area that contains the helicopter. The GET command will place this image into the memory area set aside when we dimension an array of 100 elements to hold the double-precision variable A—DIM A#(100). (The formula to determine the array size is found on page 10.61 of the Z-BASIC manual.)

We now have the first of three images that we can make appear and disappear on the screen by using the PUT command.

With the image still on screen, line 1090 adds a few pixels of length onto each side of the rotor blade. Line 1100 then saves and labels the entire modified image as B#. Line 1110 adds still more length to both sides of the rotor blade, and line 1120 saves this form of the helicopter as C#.

Now we have three images of the helicopter in memory: each looks the same except for the length of the rotor blade. What remains on the screen will be the same image we stored as C#.

Before going further, though, let's remember how the PUT command works on screen. Let's say we added

```
PUT(0,0),A#
```

as a new line, line 1082. Then, when we ran the program, a duplicate of the image would appear with its upper left corner in the screen's upper left corner (0,0). If we added that same PUT command as line 1084 and ran the program, the image would appear and then quickly disappear. In other words, if we PUT the same image to the same coordinates twice, the image disappears. A third PUT would make the image appear again, and a fourth would make it disappear—and so on, ad infinitum.

(PUT takes a third specification besides the coordinates and the array variable. This is the "action verb," which directs how the image put to the screen will interact with one already there. The default is XOR—exclusive OR—a pixel being PUT will be lit only if the current screen pixel is not lit, and turned off if it's on. If we used PUT to display the image stored in B# on top of the image taken from A#, all we would see would be the difference—the few extra pixels in the rotor blade.)

So, at this point, we have to clear the screen. We could use CLS; or we could PUT a second image C# on top of the one that remains on the screen. The latter is executed in line 1130 and causes the image to disappear. Lines 1190, 1210, and 1230 use the same technique.

Lines 1180 through 1230, then, cause each of the three images to appear and disappear in sequence. I use the same coordinates in each of the PUT commands so that the images appear in the same place on the screen each time the command is executed.

I want the images to replace each other with minimal delay, but at the same time, I want the image to remain on the screen long enough for our minds to register its existence. In the subroutine in line 1670, the delay loop allows each image to remain on the screen a fraction of a second longer than the time that would normally elapse between execution of the PUT lines. The illusion of smooth free-flowing movement depends on the maximum value of C in line 1670.

I've used a value of 75 as the maximum value of C in line 1670 so that the illusion of movement will work well both on machines running at 4.7 MHz and on those running at 8 MHz. The maximum value of C can be "tweaked" for your particular machine and preference. Increasing the value allows the image to remain on the screen longer; decreasing the value causes the image to remain on the screen for a shorter period of time.

The INKEY$ statement in line 1430 gives us a convenient method of stopping program execution. The INKEY$ function is similar to the INPUT$(X) function: it checks for input from the keyboard, but does not stop program execution while waiting for keys to be pressed. If the upper- or lowercase Z is pressed, program execution ends.

By changing the IF... THEN... portion of line 1430 to:

```
IF ZZ$<>"" THEN END
```

the program will terminate when any key is pressed.

The GOTO statement in line 1660 directs program execution back to line 1170 where the three images again appear and disappear.

By saving different images as A#, B#, and C#, (and/or D#, E#, etc.) experimenters are unlimited in the types of illusion they can create. And the illusion can be composed of any number of separate images.

For example, an animated sequence of a small man kicking a ball can be made by saving the appropriate images: the man with one leg behind him; the man standing; the man with his leg in front of him; the man with his leg in contact with a ball; and, finally, the same image with the ball a bit beyond the man's leg.

You have to operate within the limits of BASIC, however. For instance, the smaller you make the moving images, the better. Larger images do not appear on the screen all at once. They appear more like small blinds being pulled down; the illusion of motion is not as smooth.

Larger images can be used if you plan to compile the programs, since then the speed of execution is greatly increased.

Even without compiling, though, smooth-flowing motion can be created on

SEXTANT **The Independent Magazine for Users of Heath/Zenith Microcomputers**

Yes! I want to start my subscription to *Sextant* right away!

Enter my subscription for:
☐ 12 issues for $29.91 ($34.50 to Canada)
☐ 6 issues (1 year) for $14.97 ($17.25 to Canada; $21.00 overseas via surface mail, $35.00 via air mail)

Name ____________________

Address ____________________

____

____

Payment terms:
☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank.)
☐ Bill me
☐ Charge my: ☐ VISA ☐ MasterCard

Card # ____________________

Expiration date ______

We'll start your subscription with Issue #22 (May-June 1986), unless you request otherwise.

How did you obtain this copy of *Sextant*?
1. ☐ Subscription
2. ☐ Retail Store
3. ☐ Heath Store
4. ☐ Friend
5. ☐ Other

S21-S

SEXTANT **The Independent Magazine for Users of Heath/Zenith Microcomputers**

Yes! I want to start my subscription to *Sextant* right away!

Enter my subscription for:
☐ 12 issues for $29.91 ($34.50 to Canada)
☐ 6 issues (1 year) for $14.97 ($17.25 to Canada; $21.00 overseas via surface mail, $35.00 via air mail)

Name ____________________

Address ____________________

____

____

Payment terms:
☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank.)
☐ Bill me
☐ Charge my: ☐ VISA ☐ MasterCard

Card # ____________________

Expiration date ______

We'll start your subscription with Issue #22 (May-June 1986), unless you request otherwise.

How did you obtain this copy of *Sextant*?
1. ☐ Subscription
2. ☐ Retail Store
3. ☐ Heath Store
4. ☐ Friend
5. ☐ Other

S21-S

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 13098 WASHINGTON, D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

BUSS/SEXTANT
716 E Street S.E.
Washington, DC 20003-9990

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

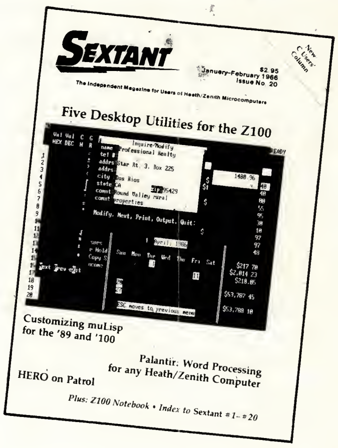
FIRST CLASS PERMIT NO. 13098 WASHINGTON, D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

BUSS/SEXTANT
716 E Street S.E.
Washington, DC 20003-9990

Complete your Heath/Zenith system with a subscription to *Sextant*!

SEXTANT



The Independent Magazine for Users of Heath/Zenith Microcomputers

Whether you use an **H/Z100, H/Z89, Z90, H88, or H8**—you'll find articles in every issue which apply to your system.

Explore **CP/M, HDOS, Z-DOS,** and **MS-DOS** capabilities in *Sextant*. Applications, programs, compatible hardware and software, and the latest developments in the Heath/Zenith community are among the topics covered in every issue of *Sextant*.

Regular *Sextant* features include:

- **"How-to" articles** to help you enhance your system. Read how other users have altered their systems to suit their needs—and how you can too.
- **Reviews of products** from Zenith and independent suppliers, which can help answer your questions about hardware and software purchases you're considering.
- Short program listings—including **utility programs and games**—which you can use immediately.
- **Advertising by independent suppliers** who support Heath/Zenith systems. In *Sextant* you'll find many products advertised that you just won't find anywhere else.
- Coverage of community affairs. You'll read about **major events and personalities** in the Heath/Zenith user community, and what effect they could have on you.

You can't afford to miss all the information packed into each issue of *Sextant*. **Your Heath/Zenith system just isn't complete without a *Sextant* subscription!**

Get the information you need to get the most from your system!

Start your subscription today!

Call Toll Free: 800/341–1522 (DATATEL 800™, for orders only, Monday-Friday 8 a.m.–9 p.m., and Saturday 9 a.m.–5 p.m., Eastern Time)

Or mail this coupon to:
Sextant, Dept. S21
716 E Street, S.E.
Washington, DC 20003
Please allow 6–8 weeks for delivery of your first issue.

Your satisfaction with *Sextant* is completely guaranteed. If at any time you're not completely satisfied, just let us know and your money will be refunded—even for the issues you've already received.

Send me:

- ☐ **12 issues for $29.91 ($34.50 to Canada)**
- ☐ **6 issues (1 year) for $14.97 ($17.25 to Canada; $21.00 overseas via surface mail, $35.00 via air mail)**
- ☐ Payment enclosed (Checks must be in U.S. dollars payable on a U.S. bank.)
- ☐ Bill me
- ☐ Charge my: ☐ VISA ☐ MasterCard

Card # ______________________ Expires ________

Name ______________________

Address ______________________

____

____

Sextant, Dept. S21, 716 E Street, S.E., Washington, DC 20003

```
1240 FOR X=1 TO 100                                   :REM 2
1250 PUT(305+2*X,105),A#:GOSUB 1670                   :REM 2
1260 PUT(305+2*X,105),A#                              :REM 2
1270 PUT(305+2*X,105),B#:GOSUB 1670                   :REM 2
1280 PUT(305+2*X,105),B#                              :REM 2
1290 PUT(305+2*X,105),C#:GOSUB 1670                   :REM 2
1300 PUT(305+2*X,105),C#                              :REM 2
-
-
1440 NEXT:END                                         :REM 2
```

Listing 2. MOTION2.BAS is created by deleting lines 1180-1230 from Listing 1 and adding the lines shown here. Under MOTION2, the helicopter will travel across the screen.

```
1310 IF ZZ$=CHR$(28) THEN A=4:D=0                     :REM 3,4
1320 IF ZZ$=CHR$(29) THEN A=-4:D=0                    :REM 3,4
-
-
1360 PUT(B+A,105),A#:GOSUB 1670                       :REM 3
1370 PUT(B+A,105),A#                                  :REM 3
1380 PUT(B+A,105),B#:GOSUB 1670                       :REM 3
1390 PUT(B+A,105),B#                                  :REM 3
1400 PUT(B+A,105),C#:GOSUB 1670                       :REM 3
1410 PUT(B+A,105),C#                                  :REM 3
1420 B=B+A                                            :REM 3
-
-
1620 IF B<10 THEN FOR C=1 TO 1000:NEXT:B=600          :REM 3,4,5,6
1630 IF B>600 THEN FOR C=1 TO 1000:NEXT:B=10          :REM 3,4,5,6
```

Listing 3. MOTION3.BAS is created by deleting lines 1240-1300 and 1440 from MOTION2.BAS and adding the lines shown here. MOTION3 enables you to control the left/right movement of the helicopter with the Z100's left/right arrow keys.

larger images if you limit the areas of motion. A full-screen image of a robot, for example, could have rolling eyes, a moving mouth, and clutching appendages.

Creating a travelling object with moving parts

With a small amount of additional code, we can make the helicopter with the moving rotor blade move across the screen. This is achieved by adding a FOR... NEXT loop to the program and by incrementing the column coordinates in each of the PUT statements.

To create MOTION2.BAS, just delete lines 1180 through 1230 from Listing 1 and add the lines given in Listing 2.

In Listing 2, line 1240 begins a loop which is executed 100 times. Each time program execution reaches the NEXT statement in line 1440, the value of X is increased by 1.

In lines 1250 through 1300, I have modified the pixel column coordinates of the PUT statements by adding two times the value of X.

The first time through the loop, the three images appear and disappear two pixels to the right of the original pixel coordinate of 305. The second time through the loop, two times X will be four; so the images will appear and disappear four pixels to the right of the original coordinates. This process continues until X reaches a value of 100 and the FOR... NEXT loop has been satisfied.

Using X and the FOR... NEXT loop causes the helicopter to move to the right of its original position a length of 200 pixels in two-pixel steps. If we subtracted two times X in each of the PUT statements, the helicopter would move 200 pixels to the left in two-pixel steps. If we changed the 2 in the PUT statements to a larger number N, the helicopter would travel N times 100 pixels to the right.

If the value of the column pixel coordinate in the PUT statements becomes greater than 639 minus the width of the image, an ILLEGAL FUNCTION CALL error message will be displayed. The same message will be displayed if the coordinate value is less than zero. An image with a width of 10 pixels must have a pixel column coordinate of less than 629 so that the entire image can be fit on the screen. (A method to get around this problem is incorporated in MOTION3.BAS.)

How can I control the movement of things on the screen with the arrow keys?

This question, too, breaks down into smaller ones: first, we can move the image left and right; then straight up and down, as well; finally, it can be moved in the diagonal directions.

After that, we can use the keypad to change the picture itself; the example I'll give here will be to drop a shell from a moving image while also controlling its direction of movement.

Moving an image with the right and left arrow keys

To create MOTION3.BAS, delete lines 1240 through 1300 and line 1440; then add the lines given in Listing 3. You'll see that I have added program lines to make the movement of the helicopter respond

to the left and right arrow keys.

The values of 28 and 29 in lines 1310 and 1320 are the ASCII-code or decimal equivalents of the right and left arrow keys. Although the arrow keys do not send a "character" to the screen, the decimal form of each is read from the keyboard when it is pressed. We can use the CHR$ function in lines 1310 and 1320 to convert the decimal equivalents back to a form that the INKEY$ function can read.

Using CHR$(90) and CHR$(122) in place of the Z and z in line 1430 would not change the effect since 90 and 122 are the decimal equivalents of Z and z respectively. The statements IF ZZ$="A" and IF ZZ$=CHR$(65) are interpreted as equivalent statements by many forms of the BASIC language.

The ASCII-code or decimal equivalents of a character can be found on ASCII charts which are included as appendices in Heath/Zenith operating-system manuals.

Another way to find the decimal equivalents is to use a program similar to that given in Listing 7 (GETASC.BAS).

This program uses Z-BASIC's ASC function to return the decimal equivalent of ZZ$. The ZZ$=INPUT$(1) statement specifies that program execution will proceed after one key is pressed. If we replaced the (1) with a (2) then program execution would proceed after two keys were pressed, one at a time.

(The same program can be used to find the decimal equivalents of most keys on the Z150 computers. However, each time a *keypad* key is pressed on the Z150, two characters are sent for processing. Of this two-character string, we need the decimal value of the right-most character, or the right-most character itself. By changing ZZ$=INPUT$(1) to ZZ$=INPUT$(2) in the program above, the right-most characters of the keypad keys are found on the Z150. Press all other keys twice each to find their decimal equivalents.)

By using the INKEY$ function, and defining characters or decimal equivalents as in lines 1310 and 1320 in Listing 3 (MOTION3.BAS), we are telling the computer to proceed with program execution until one of these keys is pressed. If none of the "inkeys" are pressed, the program keeps looping through lines 1170 to 1660.

While looping, the value of the column coordinate B+A in the PUT statements remains the same. (The value of B is equal to 305 as defined at the end of line 1130, and the value of A defaults to 0 since it is not yet defined.)

As soon as the right or left arrow key is pressed, the value of A becomes 4 or –4 according to lines 1310 and 1320. The value of the column coordinate in the PUT statements then increases or decreases by four, causing the three helicopter images to be placed at the new coordinates.

Line 1420 makes the value of B equal to B+A; the next time the value of B is plugged into the PUT statements, therefore, the images are placed four pixels to

```
1330 IF ZZ$=CHR$(31) THEN A=0:D=2                  :REM 4
1340 IF ZZ$=CHR$(30) THEN A=0:D=-2                 :REM 4
1350 IF ZZ$=CHR$(11) THEN A=0:D=0                  :REM 4
-
-
1550 PUT(B+A,E+D),A#:GOSUB 1670                    :REM 4,5,6
1560 PUT(B+A,E+D),A#                               :REM 4,5,6
1570 PUT(B+A,E+D),B#:GOSUB 1670                    :REM 4,5,6
1580 PUT(B+A,E+D),B#                               :REM 4,5,6
1590 PUT(B+A,E+D),C#:GOSUB 1670                    :REM 4,5,6
1600 PUT(B+A,E+D),C#                               :REM 4,5,6
1610 B=B+A:E=E+D                                   :REM 4,5,6
-
-
1640 IF E<10 THEN FOR C=1 TO 1000:NEXT:E=200       :REM 4,5,6
1650 IF E>200 THEN FOR C=1 TO 1000:NEXT:E=10       :REM 4,5,6
```

Listing 4. MOTION4.BAS is created by deleting lines 1360-1420 from MOTION3.BAS and adding the lines shown here. MOTION4 enables you to control the helicopter's movement in the four directions of the Z100's arrow keys.

```
1450 IF ZZ$="6" THEN A=4:D=0                       :REM 5,6
1460 IF ZZ$="4" THEN A=-4:D=0                      :REM 5,6
1470 IF ZZ$="2" THEN A=0:D=2                       :REM 5,6
1480 IF ZZ$="8" THEN A=0:D=-2                      :REM 5,6
1490 IF ZZ$="5" THEN A=0:D=0                       :REM 5,6
1500 IF ZZ$="9" THEN A=4:D=-2                      :REM 5,6
1510 IF ZZ$="7" THEN A=-4:D=-2                     :REM 5,6
1520 IF ZZ$="1" THEN A=-4:D=2                      :REM 5,6
1530 IF ZZ$="3" THEN A=4:D=2                       :REM 5,6
```

Listing 5. MOTION5.BAS is created by deleting lines 1310-1350 from MOTION4.BAS and adding the lines shown here. MOTION5 uses the numeric keypad to control movement of the helicopter in diagonal directions, as well as up/down and left/right.

```
1140 LINE(200,112)-(202,115),4,BF                  :REM 6
1150 DIM E#(20):GET(200,112)-(202,115),E#          :REM 6
1160 PUT(200,112),E#                               :REM 6
-
-
1540 IF ZZ$="0" AND E<190 THEN GOSUB 1680          :REM 6
-
-
1680 PUT(B+A,E+D),C#:FOR X=0 TO 3:FOR Y=0 TO 1     :REM 6
1690 PUT(B+A+15,E+D+8+6*X),E#                      :REM 6
1700 FOR T=1 TO 25:NEXT:NEXT:NEXT                  :REM 6
1710 PUT(B+A,E+D),C#:RETURN                        :REM 6
```

Listing 6. MOTION6.BAS is created by adding these lines to MOTION5.BAS. Besides allowing control of the helicopter's direction, MOTION6 will enable you to drop a shell from the helicopter with the zero key.

the left or right of the previous image.

That keeps the helicopter moving by four pixels at a time in the direction of the last-pressed arrow key. Since the INKEY$ is always watching for input from the keyboard, the travel direction will change each time the right or left arrow key is pressed.

I have also added two "qualification" lines for the MOTION3.BAS program.

Lines 1620 and 1630 redefine the values of B: if B is less than 10, it becomes 600; and if it is greater than 600, it becomes equal to 10. On screen, these lines cause the helicopter to disappear as it approaches the extreme right and left of the screen. The helicopter then reappears on the opposite side of the screen and continues to travel in the direction of the last-pressed arrow key.

The FOR... NEXT loops in lines 1620 and 1630 cause a slight delay in program execution. I reason that when the helicopter disappears at the left of the screen, a small period of time should pass while the helicopter "makes its way across the inside of the screen."

Increase the value of 1000 in lines 1620 and 1630 and the helicopter will be gone for a longer period before it reappears. The delay is not necessary, but it seems to add to the illusion of a small helicopter making its way around the inside and outside surfaces of the computer screen.

These lines avoid the ILLEGAL FUNCTION CALL that would appear if we tried to PUT the images to negative column coordinates or to column coordinates greater than 639 minus the width of the image.

Moving an image right, left, up, and down

Including the additional lines in Listing 4 and deleting lines 1360-1420 will produce MOTION4.BAS. This program will enable the helicopter to be directed up

and down as well as left and right with the arrow keys. With one of the new lines, also, we can cause the helicopter to "hover" by pressing the HOME key.

Lines 1330 through 1350 are the additional IF... THEN... statements that check for the Down Arrow, Up Arrow and HOME keys. The variable E was set equal to 105 in line 1130. A new variable D is now a part of the IF... THEN... statements and a part of all the PUT statements.

The value of A caused the image to be placed 4 units to the right or left of the original image position. The value of D increases or decreases the row coordinates of the PUT statements; therefore, the image is placed two pixels above or two pixels below the previous position.

When each new "checked for" key is pressed, the values for A and D are changed. When A changes to plus or minus four, lines 1310 and 1320 give D a value of 0. When D changes to plus or minus 2, lines 1330 and 1340 give A a value of 0.

If we did not change the A or D to zero, the values of both would change each time the PUT statements were executed. This would result in diagonal travel; the helicopter would go up-right, down-right, up-left, or down-left, depending upon the last key pressed in either direction.

By setting both A and D to zero when the HOME key is pressed (line 1350), the coordinates in the PUT statements are not changed; so, the helicopter hovers at the same place on screen.

There are also two new "qualification" lines in MOTION4.BAS. Line 1640 redefines the value of E to be equal to 200 if it becomes less than 10; and line 1650 sets it to be equal to 10 when it becomes greater than 200. These lines avoid the ILLEGAL FUNCTION CALL error message given if we try to place the image to a negative row coordinate or to a row coordinate greater than 224 minus the length of the image.

Lines 1640 and 1650 also provide the same delay as lines 1620 and 1630.

Using the keypad for up, down, right, left, and diagonal travel

Unfortunately, the arrow keys give us only their four directions, no diagonals. If we want to add diagonal directions, though, we can use the numeric keypad. The numeric-keypad keys send a regular character for processing when pressed, so we don't have to worry about finding decimal equivalents.

It is a simple matter to get our program to accept the keypad numbers for up, down, right, left, and four diagonal directions. This is done in MOTION5.BAS. For this program, just delete lines 1310-1350 and add the lines in Listing 5.

The keypad numbers are handled by lines 1450 through 1530 of MOTION5.BAS. The 4 and 6 direct left and right motion, while the 8 and 2 direct up and down motion. Numbers 9, 7, 1, and 3 direct the diagonal directions of up-right, up-left, down-left, and down-right respectively.

For example, when the 9 is pressed during program execution, A is given a value of 4 and D is given a value of –2. (See line 1500.) Each time the PUT statements are encountered, the column coordinates increase by four, and the row coordinate decreases by two, until a different keypad key is pressed. On screen, the helicopter images are placed four pixels over and two pixels up each time through the PUT statements.

(*Note:* If you are running GW BASIC on the Z150, be sure the NUM LCK is on when running MOTION5.BAS and MOTION6.BAS. This will configure for numbers and not the arrow-key equivalents.)

Sending shells from moving objects

MOTION6.BAS is the last modification of the program. Just add the lines given in Listing 6. This addition provides a shell dropped from the helicopter each time the 0 is pressed on the keypad.

The shell itself is created, saved, and taken off the screen in lines 1140 through 1160. The LINE command in line 1140 draws a small filled-in rectangle which is then labelled as E#; this small rectangle is the shell.

Another IF... THEN... statement is added in line 1540. This line also contains a qualification: the only time the 0 is recognized here is when it is pressed *and* the row coordinate of the helicopter position is less than 190; this turns the shell off if the helicopter is too close to the bottom of the screen. We again avoid an ILLEGAL FUNCTION CALL which would result if we tried to place the shell image at row coordinates greater than 224.

If both conditions are satisfied in line

```
10 CLS:PRINT "Program to get ASC equivalents of any key...press 'Z' to end."
20 PRINT "PRESS ANY KEY:":ZZ$=INPUT$(1):PRINT ZZ$;" =";ASC(ZZ$)
30 IF ZZ$="Z" THEN END ELSE GOTO 20
```

Listing 7. GETASC.BAS **will run on both the Z100 and Z150 to provide the decimal equivalents of keyboard keys. On the Z150, the (1) in line 20 should be changed to (2) in order to get the equivalent of the numeric-keypad keys.**

```
1030: SCREEN 2:CLS                                                    :REM 1,2,3,4,5,6

1310: IF ZZ$="M" THEN A=4:D=0                                         :REM 3,4

1320: IF ZZ$="K" THEN A=-4:D=0                                        :REM 3,4

1430 ZZ$=INKEY$:ZZ$=RIGHT$(ZZ$,1):IF ZZ$="Z" OR ZZ$="z" THEN END  :REM 3,4

1330 IF ZZ$="P" THEN D=2:A=0                                          :REM 4

1340 IF ZZ$="H" THEN D=-2:A=0                                         :REM 4

1350 IF ZZ$="G" THEN D=0:A=0                                          :REM 4
```

Table 1. These lines will enable the programs created by Listings 1-6 to run on a Z150. Simply replace the appropriately numbered lines with those given here. (Note that line 1430 needs to be replaced only in MOTION3 **and** MOTION4.**) The** NUM LCK **should be on when running** MOTION5 **and** MOTION6**; otherwise the keypad keys will be read as arrows rather than numbers.**

1540, then program execution is directed to line 1680. The PUT statement in line 1680 ensures that an image of the helicopter is present while the shell is being dropped.

Without PUTting this image on the screen, the helicopter would disappear each time the shell was dropped. This would happen because the screen is blank (although for a very short period) after each trip through the PUT statements in lines 1550 through 1600. The image of the helicopter would be gone for an even longer (and noticeable) period of time while the shell sequence was being executed.

At any given time, the position of the helicopter is given by the coordinates (B+A,E+D); we can use these coordinates to PUT the "extra" image of the helicopter to the screen.

The next part of line 1680 is:

FOR X=0 TO 3:FOR Y=0 TO 1

That can be read as: "We are going to perform the command that follows four times, and each of those four times we are going to do it twice."

The command that will be performed is the PUT command in line 1690. From the discussion above, we know that when we twice PUT the same image (E#, the shell) to the same coordinates, then the image will appear and then disappear. So line 1680 makes the image of the shell appear and disappear four times.

To make the first image of the shell appear/disappear under the center of the helicopter, I have added 15 pixels to the column coordinate and eight pixels to the row coordinate.

Looking at line 1690 again, you can see that I have also added 6*X to the row coordinate. After the first image of the shell appears/disappears, the next image appears six pixels below the first. The next time (when X=2), the shell will appear 12 pixels below the first appearance of the image, then 18 (when X=3).

In line 1700,

FOR T=1 TO 25:NEXT

allows each of the images of the shell to remain on the screen a fraction of a second longer than the time when there is no image of the shell on the screen.

The remaining part of line 1700 (NEXT:NEXT) completes the loop begun by the two FOR statements in line 1680.

The PUT statement in line 1710 is the second time we PUT image C# to the same coordinates. This makes the "extra" helicopter image disappear.

The RETURN at the end of line 1710 directs program execution back to line 1550. Once program execution is back at line 1550, it is in the INKEY$... PUT loop consisting of lines 1450 through 1530. Here the program continues to perform the task it was performing before the shell was fired, while at the same time waiting for new input or direction from the keypad.

Firing the shell in any other direction can be accomplished by adding more qualification lines and related GOSUB routines. You may want to change the shell shape to a small filled circle so that it looks the same when fired from different directions.

The only other step would be to change the increment portions of a line similar to 1690. To fire to the right, for example, line 1690 could be modified to read like this:

PUT(B+A+15*X,E+D+4),E#

Experimenters' notes

There are a number of areas where you may wish to make changes in the listings given here.

For instance, regardless of clock speed, all the delay-loop values can be experimented with to slow down or speed up the image motion according to personal taste.

By increasing the values of A and D in the IF... THEN... statements, the "travel speed" will be increased. By changing the value of A to 8 and –8, the helicopter will seem to travel twice as far each time the set of images is PUT to the screen.

You may note that the helicopter will seem to freeze each time a shell is dropped. This is because I PUT the C# image of the helicopter on the screen for the duration of the shell-dropping sequence.

The problem may be eased by lowering the maximum value of T in line 1700; this will shorten the freeze time that image C# is on the screen.

If you wish to go further, there are a number of more complete ways to solve the problem.

One way would be to PUT and replace each of the images A#, B#, and C# on the screen each time the shell changes position. PUT image A# on the screen when the image of the shell is in its first position. PUT these same images to the same coordinates again so that they disappear. Do the same with image B# and the shell in its second position, and with image C# and the shell in its third position. The rotor blade will then keep turning while the shell is being dropped.

Another method would be to create and save new images, say, D#, F#, and G#. These would be the same as A#, B#, and C# except that each would also contain an image of the shell in position one, two, or three.

Then, when the 0 is pressed, you could direct program execution to a GOSUB routine similar to lines 1550 through 1600. The only changes necessary for this new GOSUB routine would be to replace the A# with D#, B# with F#, and C# with the new image G#.

Finally, you may wish to have something other than a helicopter on the screen. The helicopter image was used in these examples because it "looks the same" for any direction of travel. If you experiment with other images, you may have to save a set of images for each direction of travel. Fish, for instance, do not usually swim backwards, so a set of images would be saved for rightward travel and a different set for leftward travel.

To really complete the job, a different set of images would have to be created for each direction the fish is permitted to travel. In this case, too, a set of PUT statements would be required for each direction. GOSUB commands at the end of the IF... THEN... sequence lines would then be used to access each "direction" set of PUT statements.

Do you have any questions about programming in BASIC? If so, send them to:

Edward A. Byrnes
BASIC Questions and Answers
412 Taylor Street
Rochester, MI 48063–4327 △

ONE MILLION DOLLAR ZENITH SOFTWARE SALE

SUDDENLY HEATH/ZENITH SOFTWARE IS MORE THAN A BARGAIN. HERE'S WHY....

Payload Computer Services has purchased over one million dollars of Zenith Software from one of America's largest corporations. With buying power like this, who else but **PAYLOAD** saves you so much money? See your Heath/Zenith Catalog for details on each item. All packages shipped complete with manuals and registration cards. **MAIL** or **PHONE** your order today and expect prompt service. **MASTERCARD** and **VISA** gladly accepted with no additional charge.

ZENITH SOFTWARE FOR THE Z-100 SERIES COMPUTERS

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MP-463-3	MailMerge for WordStar v3.21	$135.00	$ 28.00
MP-463-11	MailMerge for WordStar v3.3	$249.00	$ 85.00
MS-463-1	Z-BASIC (16-bit) interpreter	$175.00	$ 30.00
MS-463-7	Multiplan	$195.00	$ 30.00
MS-253-1	Microsoft BASIC-80 (8-bit)	$175.00	$ 30.00
OS-53-2	CP/M-85 (8-bit)	$150.00	$ 45.00
OS-63-4	Z-DOS	$150.00	$ 35.00
CB-463-7	CPS Computerized Phone System	$ 40.00	$ 20.00
CB-463-9	PECON Peachtree to Condor	$ 99.00	$ 20.00
CB-463-11	Z-Chart	$150.00	$ 22.00
CD-463-2	Condor File Manager	$299.00	$ 25.00
RS-463-1	Peachtree General Ledger	$399.00	$ 94.00
RS-463-2	Peachtree Accounts Receivable	$399.00	$ 94.00
RS-463-3	Peachtree Accounts Payable	$399.00	$ 94.00
RS-463-5	Peachtree Inventory Management	$499.00	$ 99.00
RS-463-6	Peachtree Sales Invoicing	$299.00	$ 82.00
RS-463-75	PeachText 5000	$395.00	$ 77.00
SC-463-1	SuperCalc	$195.00	$ 45.00
TA-463-1	ZDS Teacher Program	$295.00	$ 75.00
TA-463-2	ZDS Student Program	$100.00	$ 30.00
WI-463-1	Remote Batch Terminal Emulator	$899.00	$ 95.00
PO-463-1	3270 IBM Emulator	$650.00	$145.00

ZENITH SOFTWARE FOR THE Z-150/160 SERIES COMPUTERS

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
MS-5063-7	Multiplan	$195.00	$ 85.00
RS-463-75	PeachText 5000	$395.00	$ 77.00
BP-5063-1	BPI General Accounting	$595.00	$160.00
BP-5063-2	BPI Accounts Receivable	$595.00	$145.00
BP-5063-3	BPI Accounts Payable	$595.00	$145.00
BP-5063-4	BPI Payroll	$595.00	$145.00
BP-5063-5	BPI Inventory	$795.00	$180.00
BP-5063-71	BPI Series Self-Training	$ 69.00	$ 35.00
BP-5063-8	BPI Personal Accounting	$195.00	$ 75.00

ZENITH SOFTWARE FOR THE H/Z-89 SERIES COMPUTERS

PART NUMBER	DESCRIPTION	LIST PRICE	SALE PRICE
HSC-837-1	SuperCalc	$195.00	$ 60.00
HES-837-1	Microstat (96 TPI)	$249.00	$ 60.00
HMP-837-1	SuperSort	$195.00	$ 60.00
HMP-837-4	DataStar	$295.00	$ 60.00
HCP-837-1	Pearl #3 Program Generator (96 TPI, Requires C-BASIC)	$450.00	$ 60.00

ZENITH IBM COMPATIBLE COMPUTERS

ZF-158-21 One 360K floppy disk drive, 256K RAM, color ready. Retail $2199.00, Payload $1592.00

ZF-158-42 As above with two floppy disk drives and 256K RAM. Retail $2499.00, Payload $1767.00

ZW-158-42 As above but with 10.5 MEG Winchester Drive and one floppy disk drive. Retail $3699.00, Payload $2545.00

PE-158-2FW PAYLOAD ENHANCED model 158 computer. Two floppy disk drives, one 20 MEG Winchester drive and 640K RAM. Your VERY BEST BUY and a PAYLOAD EXCLUSIVE. Retail value over $4,000.00, PAYLOAD Price $2417.00

ZFL-171-42 PORTABLE Backlighted 25 line X 80 column LCD screen, 256K RAM, two ⅓ height 360K disk drives, ROM based software. Approx. 14.3 Lbs. Retail $2699.00, Payload $1889.00

All above come with MS-DOS supplied by Zenith. We also offer all premiums (if any) supplied by Zenith at time of purchase. When comparing prices, remember, Payload charges only $5.00 for shipping of prepaid orders. We can enhance the above computers with more memory or larger Winchester drives at low costs to you. All other Zenith computers sold at comparable low Payload prices. Please call for prices.

ZENITH MONITORS

ZVM-136 Long Persistance 13" High Res. $586.00
ZVM-135 Color 13" High Res., RGB & Composite ... $463.00
ZVM–133 Color 13" High Res., RGB only $437.00
ZVM-122A Amber Non-Glare 12" $119.00
ZVM-123A Green Non-Glare 12" $119.00

UPGRADE ACCESSORIES FOR Z-150/160 SERIES COMPUTERS

MEMORY UPGRADE KITS with easy to follow instructions. **MEMORY KIT #150-64-9.** Each kit will increase memory by 64K up to 320K. Only the finest 150 ns RAM chips included .... $14.50.

MEMORY KIT #150-256-18. Includes a ZPAL chip which allows use of the new 256K RAM chips included (18 pieces 256K 150 ns RAM chips). Kit increases 128K memory to 640K. If you have over 128K now, will go up to 704K. All plug into your single Zenith Memory Board. Unbelievable but true .... $94.00.

Winchester Hard Disk Drive Internal Set-up. Includes Winchester drive, controller/interface card, cables and all hardware. With 20.0 MEG (formatted) drive $592.00

Winchester Hard Disk Drive External Set-up. Includes Winchester drive, controller/interface card, power supply and case with fan. With 20.0 MEG (formatted) drive..... $797.00

First Capitol Delivers Zenith. Your Way.

Have it your way.

If you're looking for a PC, you couldn't do better than Zenith. And you couldn't buy your Zenith at a better place than First Capitol. We'll put together a system the way YOU want it. And we'll do it for a price that competes with the guy who just sells boxes that fall on his loading dock.

Great Prices. And A Great Machine.

We're committed to Zenith. And to you. the standard Zenith models that come from the factory are a good buy. And we'll be glad to sell you one at a price that will match anyone. But if you want the full potential, we'll fix it up for you. Larger disk drives. More memory. Internal tape back up. Removable winchesters. All at a budget pleasing price.

We make Zenith Scream.

When a new model comes out, we're among the first to get it. Our Zenith fanatics take it apart, prod it and poke it until it screams for mercy. Then we put it back together (usually with some new parts) so that it REALLY screams. With power.

You'll be back.

Check with us. Check elsewhere. We'll put together a custom quotation for you we GUARANTEE you'll find interesting. We think you'll come back to us. There's a reason for the growing list of companies, government agencies, and end users that know us as the source for Zenith. Your Way.

First Capitol Computer

1106 First Capitol Dr.
St. Charles, MO 63301

1-800-TO-BUY-IT (800-862-8948) Orders and quotes

1-314-946-1968 Technical support and order status

The StockSystem Challenge.

We will meet or beat any dealer's price on new stock Zenith systems. And for enhanced systems, no one can touch our combination of value, performance, and custom service.

Before you buy . . . . . .

Call Us First. And Last.

ZENITH data systems

AUTHORIZED SALES AND SERVICE

Circle #206 on Reader Service Card

A Quick Look at the Z200

Zenith's PC AT clone provides a lot of bytes for a lot of bucks.

Robert J. Gray

The IBM Personal Computer came out in 1981, and quickly set off what one publication has dubbed "The Clone Wars." Heath/Zenith has been a very successful contestant in that struggle, starting off with its first PC-compatible entry in early 1984: the H/Z150.

Neither IBM nor Heath/Zenith (nor anyone else) has stood still in the meantime.

In late 1984, IBM came out with the PC-AT. Since then, the AT-compatible market has seen some fifteen different companies enter into the latest round of the clone wars.

Zenith's AT entry, the Z200, was introduced at the COMDEX show held in Atlanta in May last year. With the Z200, Zenith has added a powerful and sophisticated state-of-the-art machine to their line of microcomputers.

I first saw the '200 shortly after COMDEX, at Navy Micro '85, the Navy's microcomputer conference held in Virginia Beach, Virginia. I had to wait until mid-September, though, before I got to borrow a pre-production model for a long weekend. The Z200 became generally available that fall.

The model I reviewed is the Z241. For some comments on the more powerful Z248, see "The 'Eight-in-One' Z248", accompanying this article.

Beyond the '150

By way of similarities with the '150, the '200 continues an important element in design philosophy: both are "backplane machines." The basic machine itself is little more than a power supply and an empty circuit board (the backplane). All other boards (processor, memory, etc., as well as add-ons) are plugged into the backplane. This philosophy facilitates expansion and designing add-on boards.

Robert J. Gray is a Commander in the U.S. Navy. He's on active duty at the Pentagon, where he's director of an office-automation and microcomputer project.

Photo courtesy of Zenith Data Systems

The '150 was a change from the "motherboard" philosophy of the 8088-based H/Z100. There (as in the older, eight-bit H/Z89), the main board has the CPU and a lot of RAM already built in.

(Interestingly, the backplane philosophy harks back to the first Heath microcomputer: the H8. The '100 encourages add-ons by incorporating the S100 bus structure: in effect, a backplane accessed by the motherboard.)

For many users, however, speed and power are at least as important as expandability. Here, the differences between the '200 and the '150 come into play.

Where the IBM PC and '150 are based on Intel's 8088 microprocessor chip, the central processing unit (CPU) in the AT and '200 is Intel's faster and more powerful 80286. Running at six or eight (or even ten) megahertz, it gains a noticeable speed increase over the earlier five-MHz machines.

Moreover, the 80286 is a "true" 16-bit chip. The 8088 is a 16-bit chip internally; externally, though, it still accesses data eight bits at a time. The 8088 can address random-access memory (RAM) of no more than a megabyte. The Z200, however, comes standard with 512K RAM and can go to more than 15 MB!

Altogether, the '200 is a very significant step up.

Looking under the cover

My first impression of the Z200 was one of understated power. It is somewhat larger than the Z150, measuring 21 inches wide by 16 inches deep and 6 inches high. The Z150, by comparison, measures 16" x 17" x 6". At 35 to 38 pounds (depending on the number and type of drives), the '200 is definitely not in the lightweight category.

Access to the insides is made easy by a cover designed to slide forward over the

front panel, once the screws on the rear and bottom edges are removed. The panel itself contains a keyed switch to disconnect the keyboard and lock the cover; the key is removable in either position. There are also light-emitting diodes (LEDs) to indicate power on and Winchester disk access.

The backplane on the Z200 has four of its slots (3, 4, 5, and 6) configured for the 128-pin Zenith "dual-plug" type of circuit boards. Slots 1, 7, 8, and 9 are 96-connector PC-AT type. The other two slots (2 and 10) take standard Z150-type circuit boards (64-pin, single-plug type). The Z200 should have no problem taking PC-type "skirted" cards.

The basic Z200 will come with only three of the ten slots filled—the CPU, disk-controller, and input/output (I/O) boards. Customers will have their choice of monochrome and color video boards, as well as additional memory and other option boards. Since nearly every configuration will have at least one video board, a realistic configuration would give you six empty slots for options.

The CPU board comes standard with 512 kilobytes of random-access memory (RAM), as well as 64K of read-only memory (ROM) for the monitor routines.

The CPU board also carries 4K of complementary metal-oxide semiconductor (CMOS) RAM used to store configuration information and calender and clock time. The backplane contains a rechargeable lithium battery used to keep this CMOS memory "alive" when the main power is off.

The backplane also has six LED indicators to aid in troubleshooting. Each one comes on as a particular voltage supplied to the computer comes within specification. The power supply is rated at 200 watts.

The last time I talked to Zenith, they were not planning to release the ROM listings. That is an unfortunate change from the Z100, but I imagine most users will not miss them.

The CPU's clock speed is six megahertz, but Zenith claims that it will work faster than the PC-AT and even units running eight-MHz clocks. The reason is that the Z200 incorporates a "no-wait-state" memory, meaning basically that the CPU does not waste cycles waiting for the memory to catch up. The "wait-state" mode may be selected, if desired, by means of a jumper.

A socket is included for the 80287 math co-processor. Power-up diagnostics are run from the ROM.

The disk controller is designed to handle two floppy and three Winchester drives at the same time. The 5¼″ floppies can be a mix of 360K and 1.2 MB, with the latter (Mitsubishi 4854) being standard. The 1.2-MB drive will read and write 360K floppies. I was cautioned that I might occasionally have a problem writing to a 360K disk, but I did not encounter any. Zenith plans to distribute its software in the 360K format.

Regrettably, the unit I had did not come with a Winchester drive, so I was unable to see how the hard disk performs. The specs for the standard drive are 20 MB formatted, with a 40-millisecond access time. There is ample space for expansion in the front of the cabinet; so the Z200 is an ideal machine for the systems integrator who wants to fit in higher capacity and/or multiple drives. (More on that later.)

The Zenith video boards are designed to plug into either of the two 64-connector "PC-type" slots in the motherboard, so the board was in slot two. The card I had was driving a monochrome monitor. The display was sharp and flicker free, and boy, was it fast!

Serial and parallel ports are standard on the I/O board. In this area, unfortunately, I feel Zenith has gone a step backwards.

The serial port is a DP-9 nine-pin connector, instead of the more common 25-pin connector. I'm sure it was done to maintain "compatibility" with the AT, but why duplicate a bad idea? What about all the people with 25-pin cables on their printers and modems? Maybe it's a clever marketing plan to sell adaptors. Fortunately, the parallel port uses the more standard DP-25.

For many users, speed and power are at least as important as expandability.

The I/O card also contains a set of six red LEDs to show the progress of the built-in diagnostics. The indicators show the status of the test of the CPU, ROM, RAM, interrupts, and disk(s) and controller; and the last one shows that a successful boot has been accomplished. If the system stops before all the LEDs are lit, the first dark one shows where the problem is.

Zenith advertises that the RAM can be expanded in 1.5-MB steps by installing Z405 memory boards. All of the RAM is implemented with 256-kilobit chips. A basic system would use slots 1 through 4, and the Z405 uses "two-connector" slots. So, you would be able to get only two Zenith add-on memory cards (in slots 5 and 6), for 3 MB, plus another 512K on the CPU board.

Zenith or someone else will have to come up with higher density memory boards to take full advantage of the 16-MB address size of the 80286. Another option is to use PC-AT memory boards in slots 7, 8, and 9—which would probably require setting the CPU to enable wait states.

On the other hand, how many of us are really going to need more than eight megs of memory? Remember how much we used to get done in 48K?

Conversation with an engineer friend of mine who does memory design for a living turned up an interesting consequence of going to multi-megabyte memories. It seems that the memory arrays (working under five volts DC) draw about 500% more current on power up than they do during steady state. Failure to have an adequate power supply could result in problems. Zenith, fortunately, is a recognized leader in power-supply design, and their supplies are usually much heftier than the competition.

At first glance, the keyboard on the '200 appears identical to that on the Z150. You have to look closely to find that a new key has been added—labelled "System Request", and located in the top right-hand corner.

All of the Z150's keys can be found on the Z200 keyboard. A few, though, such as Plus, Minus, and Print Screen have been moved around a bit. The keyboard connects via a coiled cord to a five-connector plug on the center rear of the unit. Personally, I'd rather see the plug in the front, but perhaps I'm being too fussy.

Checking out the basics

Well, with my hardware survey more or less completed, the time came to finally turn the system on.

The first thing you'll notice on power up is that the cooling fan is a little loud. It's not exactly a 747 on takeoff, but it's definitely there. But with that aside, the Z200 really comes with some great features.

Zenith has done a superb job with the monitor routines and diagnostics. The system I had was running version 0.4 of the MFM-200 monitor code. The Z200 has kept the PC's procedure of doing a system reset with CTRL-ALT-DEL. Access to the monitor routines is gained by CTRL-ALT-INS, or by hitting ESC during the boot sequence. Another sequence, CTRL-ALT-RETURN, will stop the system and display the status of the CPU and the contents of its registers.

The monitor routines have a lot of power built into them, particularly for people who work in assembler. There are the usual commands to examine and alter memory, search, etc. Other nice features include a hexadecimal math calculator: enter two hex numbers and get the sum and difference. There are also a color-bar generator, an unassembler, and a selection for type of video scrolling.

I'd like to say that all of this worked the first time out, but I ran afoul of an undocumented feature.

The technical reference manual I had was a draft copy; it did not explain the setting of something called the "video mode." So, the first time I turned it on I

got … a blank screen. A couple of phone calls later, I discovered I needed to type VM7 to set up the system for the monitor. Under normal conditions, this only needs to be done once; after that, the mode is saved in the CMOS RAM.

With video mode out of the way, I was able to get into the monitor.

In addition to the features mentioned above, you can also run a set of extended diagnostics, testing the disks, keyboard, memory, etc. The SETUP section is where you start to get a feel for the power of the Z200. This section allows you to enter date and time, and even lets you do a quick switch to and from daylight savings time.

SETUP is also where you tell the system the number and type of disks you have. For the Winchesters, you have 15 preprogrammed sets of disk characteristics, for such things as number of cylinders and tracks, precompensation, and capacity.

I found the disk-capacity options to be very interesting—because the choices ranged up to disks with 112 megabytes! Someone at Zenith is planning ahead. The system will accept a total of three Winchester drives.

The '200 was provided with version 3.10 of the Microsoft Disk Operating System (MS-DOS), and this was the only thing I had that was not a prototype. The manual comes in a three-ring binder in the now industry-standard slip-case box, and appears to live up to the Heath/Zenith reputation for excellent documentation.

There also appear to be some Zenith enhancements (â la Z100 and Z150), such as an encryption utility (CIPHER) and a utility to read CP/M disks (RDCPM). I just wish that I'd had more time to work with everything.

Zenith has also announced that they will offer Xenix, a Unix look-alike from Microsoft, as an alternative operating system. The distribution media for Xenix will be the 1.2-MB floppy format.

Sampling its power

With the possible exception of all of that memory, probably the biggest reason that users will be attracted to the Z200 is its speed increase over the 8088/8086 class of machines. I wrote a couple of test routines in GW BASIC, and ran them on both the Z200 and an IBM PC-XT. (The '150 was at the office.)

The first routine was just a simple FOR… NEXT loop in BASIC. Averaging over several trials with different numbers of iterations, the Z200 was four times as fast as the PC. The second test was designed to test more of the system, by calculating a series of square roots. Here, the Z200 averaged out to be almost ten times as fast.

If you do a lot of number crunching or heavy spreadsheet work, the Z200's combination of speed and memory is going to give you some real improvements.

How compatible is the Z200 with the Z150? I tested it with GW BASIC version 1, dBASE II, and MultiMate version 3.22—all of which ran without a hitch. I did run into an "error loading driver" problem with Lotus, but it was probably due to trying to run an executable version INSTALLed for another system, instead of starting with the distribution disks.

Overall, the Z200 more than lives up to its billing as a system for the advanced user who needs speed and lots of memory, as well as large disk capacity. I predict that it will become a favorite with system integrators, who will find room in the chassis and on the backplane to add lots of goodies. (One area, for example, where the '200's power and adaptability can be utilized is as a file server for a network of micros, or as a central "host" to which several terminals could be attached.)

So, how much does all this wonderfulness cost?

The standard version, with 512K memory, a single 1.2-MB floppy drive, and MS-DOS, lists for $3,999. Adding the 20-MB Winchester will boost the cost to $5,599. Add to that your choice of monitor and video controller card, and you can see that it's not the micro for everyone.

But if you need speed, large memory, and *real* mass storage, then the Z200 is for you.

Zenith plans on regular improvements to the Z200, to keep it current with new technology and to ensure that users continue to get maximum performance. They've sure made a great start. △

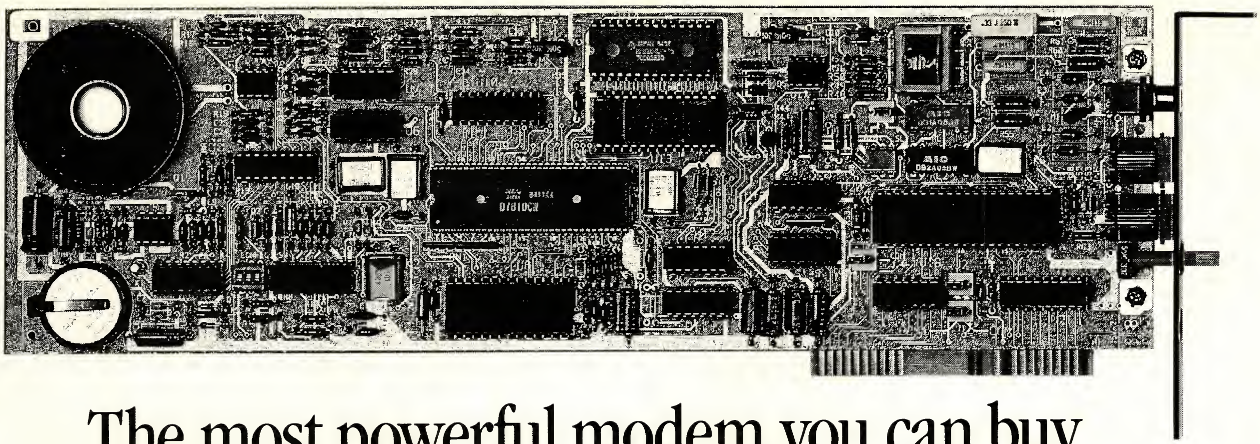
The "Eight-in-One" Z248

As this article was being prepared, Zenith Data Systems announced the newest configuration of the Z200, the Z248 "Eight-in-One" Turbo PC. It's designed apparently as an enhancement of the Z241. The Z248 features an eight-MHz clock and an increase in ROM address space from 64K to 128K. There will be a variety of standard system configurations—including a 40-MB, 40-millisecond Winchester drive (the ZD400) and a 20-MB external streaming-tape drive for backup (the Z427-20).

The Eight-in-One name apparently comes from the Z248's ability to be compatible with a variety of personal computers. The list includes the IBM's PC series and the PC with Enhanced Graphics Adaptor. Compatibility with the Z100 and Z150 will be provided by the Z319 and Z409 boards, respectively.

The Z248 is physically similar to the Z241, both internally and externally. However, the faster speed and larger disk capacity will make it ideal for multi-tasking and multi-user environments.

Initial estimates from Zenith are for availability this spring. Prices are expected to range from about $4,500 to about $7,000 depending on configuration.



The most powerful modem you can buy for your Heath/Zenith PC

The ZOOM/MODEM PC1200

If you're looking for the most advanced 1200 baud modem for your Heath/Zenith PC, look no further. You've just found it.

Of course the PC1200 is auto-dial/auto-answer with full duplex operation and 100% Hayes compatibility. That's where we started. Then we added lots of features the competition never even thought of, like Demon Dialing, call progress tone detection, access security, voice response capability, a clock/calendar, and an electronic mailbox. And to make it all easy to use, ZOOM/DISK Software.

Hayes compatible.

Why deny it. Hayes wrote the *old* standard. So before we added all our special features we made sure that the ZOOM/MODEM would do everything that Hayes would do. Respond to *every* command, work with *every* program, in *every* machine–just like Hayes.

Demon dialing.

We invented the Demon Dialer back in 1981, and gave it the ability to "Demon Dial" a busy number over and over rapid-fire until the busy changes to a ring. The ZOOM/MODEM PC1200 uses advanced tone detection for the best Demon Dialing ever. It takes the frustration and wasted time out of reaching a busy bulletin board, getting Springsteen tickets . . .

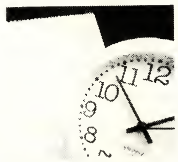
Security.

The PC1200 offers you three levels of password security. It lets you use normal ASCII passwords, touchtone passwords to guard against hackers searching for a modem tone, or a combination of ASCII characters and touchtones for maximum security. And when using automatic callback to protect against stolen passwords, our dialtone detection gives your system an extra level of security.

Voice response.

Only the PC1200 has a touchtone detector, and an on-board audio input jack that lets you plug in a voice synthesizer, microphone, or another sound source. This means that someone can call your computer from any touchtone phone and use a touchtone code to get inventory information, order status, or the latest price quotations. Plug in a microphone, and your computer becomes a remote security monitor.

Clock/Calendar.

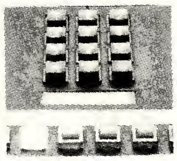


Don't you wish *all* your files had the time and date on them? That's why we put a real-time clock/calendar on board. It will load your computer's system clock every time you turn your computer on, even if you're not using the modem, so that every file will be stamped with the date and time. And a clock/calendar gives you the hardware capability to automatically access databases or send files at a pre-selected date and time, when rates are low.

Electronic mail.

If you're using your computer and someone calls in, our on-board 8K byte RAM lets you receive messages or small files in the background, while you continue working.

Office ready.



ZOOM's been a leading supplier to the telecommunications industry since 1977. So it makes sense that we'd design a modem that fits right into the office environment. That means light control for key telephones; complete instructions for office installation, and knowledgeable technical support.

And more.

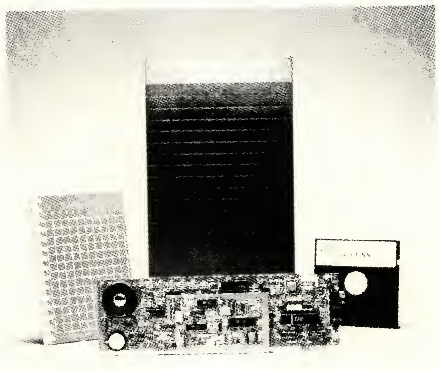
Add COM 1, 2, 3, and 4 capability, a high speed 16450 UART to keep up with the new generation of AT compatible computers, a call progress speaker with volume control, EPROM upgradeability, and software that supports all our advanced features. Now you've got it. The most powerful modem you can buy for the IBM PC.

American made.

The PC1200 is made in the USA, using sophisticated automated assembly and testing to insure high quality and trouble free performance. It's FCC approved and comes with a 2 year warranty, backed by ZOOM's excellent customer service and technical support.

Start communicating.

So stop waiting and start communicating. You won't find a better modem at any price.



ZOOM

207 South Street, Boston, MA 02111

Available at all Heath/Zenith Computers & Electronics Centers Nationwide

Circle #195 on Reader Service Card

Speeding Up Your '100 or '150 With the 8087 Math Chip

Three boards, software support, and a number of books and articles are reviewed here.

Alan T. Moffet

How would you like to speed up your scientific, graphic, or engineering computations by a factor of 20 or so?

You *could* do it by moving your programs to a mainframe computer—but there's another way that is available to owners of H/Z100 and H/Z150 series computers. Install a numeric co-processor and bring near-mainframe power into your own den or office. As the guy in the TV commercial says, "Aw-w-w-w-some!" (The Z-machine he's referring to, though, is four-wheeled and comes from Nissan.)

In this article, I'll explain how to enhance the power of your computer in this way; I'll review the supplementary boards which are available for installing a numeric co-processor in H/Z100s; and I'll describe some of the software which can make use of the co-processor. Finally, I'll give some benchmarks to show its speed.

What is it, and how does it work?

What is this thing called a numeric co-processor? It's often called a numeric data processor (NDP). It's a companion to the 8088 microprocessor, which is the heart of the 16-bit side of the H/Z100 and also of the H/Z150 (and the IBM Personal Computer and the '150's other cousins).

The 8088's co-processor is called an 8087, and it was designed by Intel to handle numeric operations with floating-point numbers, including the calculation of non-algebraic functions such as cosines and logarithms. It also can speed up operations on long (32- and 64-bit) integers and do decimal arithmetic with 18-digit precision.

(The equivalent co-processor for the 80286 central processor unit (CPU) used in the Z200 is the 80287. In these machines, patterned after the IBM AT, the 80287 works very much like the 8087 in an 8088-based microcomputer. However, the effective computation speed of the '287 is a bit *less* than that of the 8087 in a '100 or '150.)

The 8088 is a general-purpose microprocessor which can do many different arithmetic and logical operations; however, it operates only on single-byte characters and on integers with lengths of 16 bits or less. Most scientific or engineering calculations require greater precision and range than can be obtained with 16-bit, or even 32-bit, integers. These programs use floating-point numbers. (See "Floating-Point Numbers," accompanying this article, for some of the whys and wherefores of these numbers.)

The standard floating-point format used by the 8087 has been defined by the Institute of Electrical and Electronics Engineers. In this IEEE format, numbers are either 32 bits (four bytes) or 64 bits (eight bytes) in length.

The short (four-byte) numbers have a precision of 23 bits, or about one part in eight million; they can express positive or negative numbers with magnitudes as great as 2^{128}, or about 10^{38}.

The long (eight-byte) numbers have a precision of 52 bits, or one part in 4×10^{15}; they may have magnitudes as large as 2^{1024}, or about 10^{308}. (Yes, that's 1 followed by 308 zeroes, which is sufficient even for cosmological calculations!)

In a microcomputer without a numeric co-processor, computations using floating-point numbers must be done with lengthy subroutines. These subroutines use the microprocessor's integer and logical operations to break up the floating-point numbers into short integer fields which the microprocessor can handle. All of this takes a lot of instructions, and this *software floating-point emulation* usually runs very slowly.

A numeric co-processor has special arithmetic logic designed to handle floating-point operations rapidly—sort of "hardware subroutines." The 8087 also has subroutines built into it to speed up the calculation of logarithms, exponentials, and trigonometric functions. As well, it has some commonly used conversion constants such as *pi*, 0, 1, the binary logarithms of 10 and *e*, and the decimal and natural logarithms of 2.

Rounding out its capabilities are routines to convert 18-digit decimal numbers to and from floating-point format. Inside the 8087, all calculations are done in a special *80-bit* floating-point format. This allows sequential calculations to be done without the accumulation of errors due to truncation or rounding.

The tricky part of using a system with two processors is the coordination of tasks. If you have a numeric co-processor, how does it know when to take over and which numbers to process?

There are several ways to do this. The Intel designers chose to have the 8087 "read the mail" on the 8088's data/address bus. It keeps track of the flow of instructions in the program, ignoring those intended for the 8088.

However, when an instruction calling for a floating-point operation comes along, the 8087 wakes up and takes over. Then, the 8088 either idles or (if the program designer has been *very* clever) it may go on executing subsequent non-floating-point instructions while the 8087 chews away on the big numbers.

Further details on how this scheme is made to work correctly may be found in the references accompanying the article, especially in the books by Bradley and by

---

Alan T. Moffet is a professor of radio astronomy at Cal Tech. He wrote about ALTCHAR.SYS *in the March-April 1985 issue of* Sextant.

Palmer and Morse.

The 8087 does not function as a co-processor to the 8-bit 8085 CPU, so it will not speed up any computations done on that side of the '100. There are several 8-bit floating-point co-processors which can work with the 8085, but no one seems to have designed a board to permit one of these to work in the '100.

(Rick Lutowski has reported on the performance of the 8088/8087 combination in the '100, comparing it to a Z80 CPU with the AM9511 co-processor designed by Advanced Micro-Devices; the AM9511 was on an H8 board available from Trionyx. See "A Critical Look at the 8088," in *Sextant* #15, May-June 1985.)

So, to take advantage of the 8087, you must install it in your computer, and you must have programs which call it into action. First, let's deal with the installation.

Installing an 8087

This ought to be a trivial matter, and indeed on the H/Z150 (and the IBM Personal Computer, etc.), it is. The designers of these machines put a socket on the CPU board, right next to the one for the 8088, with all the circuits in place to connect up the 8087. For the H/Z150, Heath/Zenith offers the 8087 chip (the Z-316) for $225, but similar 8087 products are available from many other sources for as little as $100.

All you have to do is buy one of these and plug it in. (Just observe the precautions against damage by static discharge which I describe later on.) Installation of an 80287 in a Z200 is essentially the same.

How about the H/Z100? Unfortunately, the engineers who designed this one weren't so farsighted, and they left no space for an 8087 on the processor board. (They built in lots of other good things that the IBM designers either left out or arranged to be sold as extra-cost options, but they dropped the ball on the 8087.)

Recognizing the need many H/Z100 owners have for a numeric co-processor, three vendors, including Heath/Zenith, have brought out special boards which permit the installation of an 8087 in these machines.

Two of these, the 2+2 from D.E.L. Professional Systems (now called Gemini Technologies, Inc.) and the Z-216 from Heath/Zenith, are *piggyback boards*. They mount directly on the H/Z100's main printed-circuit (PC) board, connecting to it via a plug that goes into the original 8088 socket. The 8087 and the original 8088, plus a couple of support chips, plug into the piggyback board.

The third, the Z-Support Board from Hudson and Associates, takes a very different approach. It is an S-100 board which plugs into the H/Z100's card cage; it offers up to 256 kilobytes of expansion memory in addition to a home for the 8087 (and the transplanted 8088).

The three boards are shown in Figures 1, 2 and 3.

Why all this complexity, when all you have to do on the Z150 is plug in the 8087?

The answer is that Intel designed the 8088 to support two modes of operation. In the *minimum* mode there is no co-processor, and the 8088 itself generates certain bus-control signals. These are similar enough to those generated by the 8-bit 8085 that it is easy to let the two microprocessors share the same address and data buses and memory.

The *maximum* mode is turned on by connecting pin 33 of the 8088 to +5 volts instead of ground; then eight other pins change their functions so that the 8087 may work with the 8088. In Intel's conception of this arrangement, an 8288 bus controller is then also needed to generate

Floating-Point Numbers

An ordinary binary integer of length N bits can express numbers ranging from -2^{N-1} to $+2^{N-1}-1$; this means that the 16-bit integers most readily used by the 8088 can only express numbers from −32768 to +32767.

That range is inadequate for many purposes; for instance, if a 16-bit integer quantity expressed a sum of money in cents, the maximum value it could hold would be $327.67, which would not be enough even to sum up your monthly bills.

Greater range may be obtained by using longer integers. For instance, 32-bit integers can represent numbers with magnitudes up to 2,147,483,651.

Scientific and engineering calculations often involve numbers with far greater range. The distance to the nearest star is about 4.04×10^{18} centimeters. (By scaling to different units, it may be expressed as 4.31 light years.) This is a number which would require 63 bits if it were expressed in integer form, and many examples which are even more extreme may readily be imagined.

However, we don't usually know such very large (or very small) numbers with an accuracy commensurate with their magnitude. The uncertainty in the distance to that nearest star is about 1%; so the distance can be adequately expressed with many fewer than 19 decimal (or 63 binary) digits.

For such a number we commonly use *scientific notation*, expressing it as a *sign*, a *power* of some *base*, which gives the order of magnitude of the quantity, and a *mantissa* which is a multiplier containing a sufficient number of digits to express the quantity to its known precision. The mantissa is usually *normalized* so that it lies in the range of 1.0 to B, where B is the base.

Thus in that stellar distance, the sign is positive (implied by the lack of a minus sign), the mantissa is 4.04, the base is 10, and the power is 18. If we convert to a base of 2 we would have $4.04 \times 10^{18} = 1.75 \times 2^{61}$.

Going to very small numbers, we can express the mass of the electron in grams as $9.1083 \times 10^{-28} = 1.12755 \times 2^{-90}$.

In most computers, the base of floating-point numbers is chosen to be 2, rather than 10. A notable exception is the IBM 360/370 series, which uses a base of 16. This choice makes normalization much more tricky and makes the effective precision three bits less than the length of the mantissa. (I leave the explanation of this last statement as an exercise for the curious reader.)

The range and precision of binary floating-point numbers are set by the number of bits in the fields representing the *exponent* (the power to which the base of 2 is raised) and the mantissa, respectively.

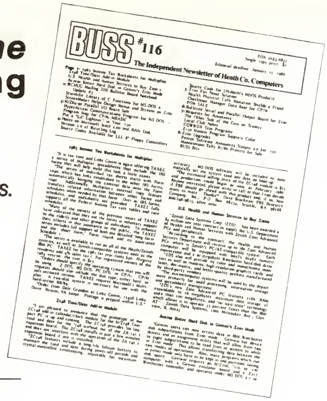
For 32-bit numbers, the most common choice is one bit for the sign, eight bits for the exponent (seven bits, plus one to allow both positive and negative powers of the base), and 23 bits for the mantissa. Since it is known that the normalized mantissa will lie between 1 and 1.999..., rather than between 0 and 1.999..., one bit of its value can be implicit. The precision is thus 24 bits, or about one part in 16,000,000. The exponent can be $\pm 2^7$, or ± 128, so the range is $2^{\pm 128}$, or about $10^{\pm 38.5}$.

In the IEEE long (64-bit) floating-point format, the exponent uses 11 bits and the mantissa 52. This yields a precision of about one part in 10^{16} and a range of $10^{\pm 108}$.

The older Microsoft 64-bit format, used up to now in all the Microsoft BASIC interpreters and compilers, uses only an 8-bit exponent and has a 55-bit mantissa.

The 32-bit and 64-bit Microsoft floating-point number formats are both incompatible with the IEEE standard used in the 8087. The 32-bit formats differ only in the placement of the sign bit, but that is enough to make them incompatible.

YES! Rush me my FREE copy of *The Buss Directory* as my bonus for trying *Buss* newsletter.



☐ Bill me $28 ($40 overseas) for 20 issues (one year) of *Buss*.

☐ I prefer to pay now.

☐ My check is enclosed (checks must be in U.S. dollars payable on a U.S. bank).

☐ Charge my: ☐ VISA ☐ MasterCard ☐ CHOICE

Card # ________________________

Expiration date ____________

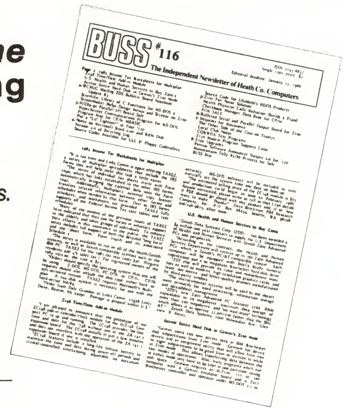
Name ________________________

Address ________________________

Buss, Dept. S21
716 E Street, S.E.
Washington, DC 20003

S21-B

YES! Rush me my FREE copy of *The Buss Directory* as my bonus for trying *Buss* newsletter.



☐ Bill me $28 ($40 overseas) for 20 issues (one year) of *Buss*.

☐ I prefer to pay now.

☐ My check is enclosed (checks must be in U.S. dollars payable on a U.S. bank).

☐ Charge my: ☐ VISA ☐ MasterCard ☐ CHOICE

Card # ________________________

Expiration date ____________

Name ________________________

Address ________________________

Buss, Dept. S21
716 E Street, S.E.
Washington, DC 20003

S21-B

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 13098 WASHINGTON, D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

BUSS/SEXTANT
716 E Street S.E.
Washington, DC 20003

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 13098 WASHINGTON, D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

BUSS/SEXTANT
716 E Street S.E.
Washington, DC 20003

Get a FREE copy of *The Buss Directory* just for trying *Buss!*

We want to send you a FREE copy of *The Buss Directory* as your bonus for trying *Buss: The Independent Newsletter of Heath Co. Computers.* Even if you decide not to take *Buss,* the Directory is yours to keep!

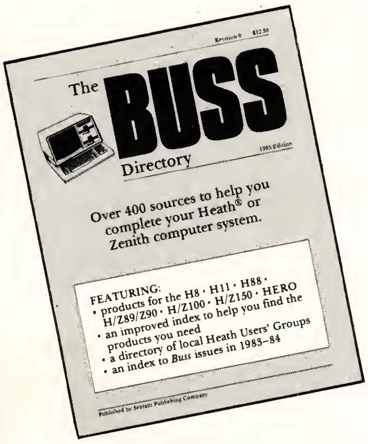
Here's how this special introductory offer works:

- We'll send you two issues of *Buss* to examine. If you find *Buss* doesn't meet your needs, just write "cancel" across the invoice you'll get. You'll have paid nothing, and *The Buss Directory* and the two issues of *Buss* you've received are yours to keep.
- If you do subscribe and ever become dissatisfied with *Buss,* simply cancel. You'll get a prompt, 100% refund of every cent you paid—no questions asked. *The Buss Directory* and all the issues of *Buss* you'll have received are yours to keep.

You can't lose with this special offer, so send in your order today to start receiving *Buss*! See what the information in *Buss* can do for you and your Heath/Zenith system.

Twenty times a year, you'll get a complete report on the latest news about your Heath/Zenith system—delivered directly to you by first class mail. *Buss* helps you:

- Learn what new products are being developed, and where you can get them.
- Benefit from the advice of users when you write in to *Buss*'s "Requests for Assistance" column.
- Save money when you add to your system! *Buss* offers a free "For Sale" listing for Heath/Zenith users where you can pick up a bargain.
- Avoid bugs in your system as you read how other users have solved similar problems with their computers.
- Discover news and events in the Heath/Zenith community that affect you.
- Meet other users in your area through the "Local Club Notes" section of *Buss.*
- Get unbiased coverage of your system from the oldest independent newsletter for Heath/Zenith users. Since 1977, *Buss* has been committed to supplying information exclusively for Heath/Zenith users.



And in addition to the support you'll find in *Buss,* you'll receive FREE the latest edition of *The Buss Directory* (normally $12.50), a reference tool with information on over 400 independent sources of support for your Heath/Zenith system. It includes hardware, software, peripheral products, and supplies you can use to complete your computer system.

We'll mail you the Directory as soon as we receive your order.

Call Toll Free: 800/341-1522

(DATATEL 800™, for orders only, Monday-Friday 8 a.m.–9 p.m., and Saturday 9 a.m.–5 p.m., Eastern Time)

Or use this coupon, and mail your order to:
Buss, Dept. S21
716 E Street, S.E.
Washington, D.C. 20003
202/544-0900

Remember, *your satisfaction with* Buss *is completely guaranteed.* If at any time you are not satisfied, any money you've paid will be refunded in full.

Yes! Rush me my FREE copy of *The Buss Directory* as my bonus for trying *Buss* newsletter.

☐ Bill me $28 for 20 issues (one year) of *Buss* ($40 overseas).

☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank).

☐ Charge my: ☐ Visa ☐ MasterCard

Name________________

Address ________________

Card # ________________

Expiration date________

Buss, Dept. S21, 716 E Street, S.E., Washington, DC 20003

Circle #107 on Reader Service Card

the bus control signals.

In the design of the '100, with its 8085 and 8088 combination, it was easier, and slightly cheaper, to use the 8088 in its minimum mode. The three 8087 boards for the '100 add either an 8288 or a custom-designed substitute to allow the 8088 to run in the maximum mode.

Installing any of the three 8087 boards requires you to disassemble the upper section of your computer. First of all, disconnect the main power cord. Then remove the top cover and remove the necessary screws so that you can lift out the disk drives, the base cover, and the keyboard (and the cathode-ray tube in the '120).

The video board must be removed, since it sits above the 8088's socket. Then the 8088 is gingerly pried up, either with a small screwdriver padded with tape or, if you should happen to have one, with an extractor tool for 40-pin integrated circuit (IC) chips.

During this, and all other operations in which you have an IC out of its socket, take precautions to avoid damaging the IC with an electrostatic discharge. Be sure to touch the metal base of the computer before touching any of the PC boards. If possible, keep a wrist or an elbow in touch with the base as you work. When you extract an IC or a board from the computer, transfer it to a metal sheet (I use a cookie tin) while you work on it, and maintain contact between your arm and the sheet as you work.

After the 8088 is removed, straighten any of its pins which may have been bent and plug it into the empty 40-pin socket on the new board. Be sure that all the pins are correctly seated in the socket. It is easy to overlook a turned-under pin.

When you are sure that everything is properly in place, you are ready to put things back together.

If you have a D.E.L. 2+2 or a Z-216, the piggyback board plugs into the vacant socket on the '100's main PC board. The Z-216 has two additional support posts to fix it in place, but the 2+2 seems reasonably secure with just the 40-pin plug holding it down.

The Hudson board goes into the S-100 card cage. The connection between the board and the original 8088 socket is by way of a short piece of 40-wire ribbon cable; this must be threaded under the metal card-support cage surrounding the S-100 area. Four small screws holding the cage in place must be loosened to permit this.

The 8087 is a "hot chip," literally as well as figuratively. It needs plenty of cooling, so make sure that nothing obstructs the flow of air around it.

In no case should installation take more than an hour or so. The biggest chore is removing the cover-latching slides in the all-in-one model, if you have it. All the tools you need are a Phillips screwdriver, a quarter-inch nut driver, and a small conventional screwdriver.

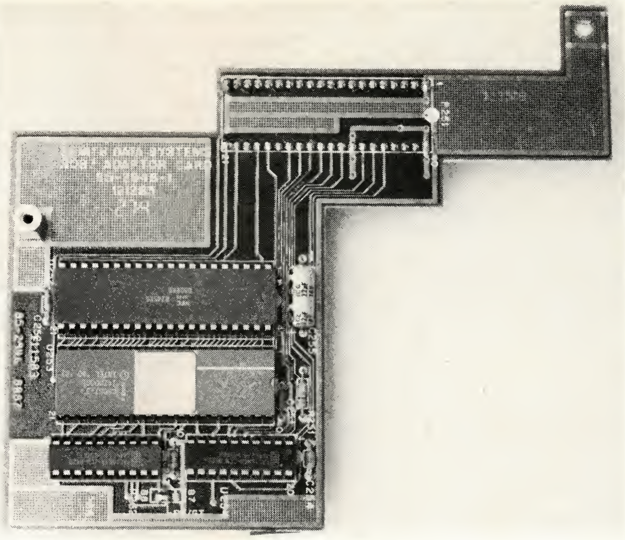


Photo courtesy of the author

Figure 1. The Z-216 board from Heath/Zenith.

Possible difficulties

When things are all together again, and the external connections have been plugged in at the back, try booting up with both 8-bit and 16-bit system disks. Run some of your favorite programs to assure that the processors all work together.

There have been reports of some 8088s and 8085s which would not work reliably with an 8087 co-processor installed. These have usually been "second source" ICs; Heath/Zenith has shipped some of the '100s with microprocessors from NEC or from Advanced Micro-Devices. Incompatibility, when it occurs, is usually with these parts. Hudson states flatly that the 8087 will not work with an 8088 made by NEC. You should take note of the manufacturer's name and part number on both the 8088 and the 8085 while you have the video board out.

For whatever it's worth, one of the two machines in which I installed 8087s has NEC chips on both the 8-bit and 16-bit sides, and it seems to work fine.

You may have installed, or plan to install, a new clock crystal to speed up your computer. If so, you should be aware that the 8087-3 which is supplied with each of these boards will almost certainly *not* work at a clock rate higher than 5 megahertz. If you are able to switch back and forth between the higher clock rate (usually 7.5 MHz) and the original 5-MHz clock, you can install an 8087-3. However, you must switch to the normal clock before running any program which makes use of the 8087-3.

Intel makes a high-speed 8087, the 8087-2, which is guaranteed to go at 8 MHz. This is a new design, made with smaller-sized transistors and interconnections, so that it is intrinsically faster. (The rumor that the 8087-2 is just a selected 8087-3 which happens to work at higher speed is not true.) D.E.L. and Hudson will sell you a board without an 8087 if you want to procure the faster unit and install it on your own. The price for an 8087-2 at this writing is about $150. Heath/Zenith sells an 8-MHz Z-216-8 for $429.

Intel also makes an 8087-6, which costs

Acknowledgements

The Heath Company and D.E.L. Professional Systems each loaned the author one of their 8087 boards for review purposes. Also, Borland International provided a review copy of Turbo Pascal.

The DeSmet C benchmarks given in Table 1 were run by Robert Logan.

a bit less and is slightly slower than the 8087-3. It is guaranteed only for the IBM PC's 4.77-MHz clock rate and may not work with the '100's 5-MHz clock. You're better off to spend a few dollars extra and get the 8087-3, at least.

Now I'll describe the features of the three 8087 boards for the H/Z100 in greater detail.

The Z-216

I'll start with the Z-216, since it is the "official" 8087 support board from Heath/Zenith. It lists for $299 and is available by mail order and at the Heath/Zenith Computers & Electronics stores.

The board, seen in Figure 1, has the 8087, a socket for the 8088, and two custom programmable array logic (PAL) chips; these two PALs take the place of the 8288 bus controller and provide all the other logic support required for the two microprocessors.

The peculiar L shape (Idaho shape?) of the board is dictated by the fact that the 8088 socket on the '100 lies *under* the rear edge of the video board. Thus it is necessary to extend the Z-216 rearwards to bring its four chips into the open space behind the video board. The Z-216 is held in position by two screws, a rubber spacer pad, and the 40 pins with which it connects to the original 8088 socket; it is quite secure mechanically.

The Z-216 comes with a 26-page installation manual containing step-by-step instructions, in the usual Heathkit style with many pictorial diagrams. The manual explains very clearly how to disassemble either model of the '100 so that the 8088 may be extracted and the piggyback board installed in its place; then it tells you how to put things all back together again.

The manual contains a circuit diagram of the Z-216 and a terse, one-page description of how the two PALs are used to synthesize the minimum-mode bus-control signals. This description is not comprehensible unless you have at hand the Intel manuals for the 8088 and 8087. (For those who enjoy Boolean algebra, the Heath/Zenith manual gives the PAL design equations.)

Heath/Zenith is the only one of the three vendors to include any information about the *use* of the 8087. In the installation manual, there is a brief example of an assembly-language subroutine for Z-BASIC, covering the same ground as a part of Appendix E in the Z-BASIC manual. Heath/Zenith also supplies with the Z-216 a copy of a paperback book, *8087 Applications and Programming for the IBM PC and other PC's*, by Richard Startz. (Later on, I'll have some more to say about this and other related books.)

The Z-216 manual fails to mention an incompatibility between the Z-216 and Z-BASIC. Z-BASIC uses the old Microsoft floating-point number format (which is mentioned in "Floating-Point Numbers," accompanying this article). The 8087, however, requires the IEEE format. Therefore cumbersome back-and forth conversions must be done if the 8087 is to be used with Z-BASIC. (The same is true for GW-BASIC on the Z150, and for the Microsoft BASIC compiler, BASCOM, on either computer.) Assembly-language routines to do these conversions are given in Startz's book.

The D.E.L. 2+2

Outwardly, the D.E.L. 2+2 board very much resembles the Z-216, as can be seen by comparing Figures 1 and 2. It has to fit in the same space as the Z-216, so it has much the same shape, although it's a little smaller.

The 2+2 board is intended to be held in place only by the 40-pin plug in the original 8088 socket. There is a hole in the 2+2 board which would allow it to be secured to the video board with a threaded stud, but no such hardware is provided. The board is very light, and it should be reasonably secure without additional mechanical support.

The 2+2 comes with a 33-page manual (half-size, 8½- by 5½-inch pages) which describes the installation process and gives the 2+2's circuit diagram. The circuit consists of the 8288 bus controller and two PALs incorporating all the additional logic.

The manual is not quite as elegant as that for the Z-216, but it is certainly adequate as far as installation goes. Several compilers for different programming languages which can take advantage of the 8087 are mentioned.

List price for the 2+2 is $299, and it is available by mail order and at selected Heath/Zenith Computers & Electronics stores and some other retailers. You may find that some dealers have boards in stock with an 8087-3 (5 MHz) installed, but the manufacturer now sells the board with an empty socket, allowing you to choose the co-processor chip to suit your computer's clock rate and to take advantage of the declining prices on these chips.

D.E.L. Professional Systems also makes the Gemini board, which allows the H/Z100 to emulate the IBM PC, and an 8087 socket is provided on the Gemini. The 8087 operates in both the IBM-emulation mode and in the native Z100 mode.

The Hudson Z-Support Board

As can be seen in Figure 3, this board is very different from the other two. It is an S-100 board which provides a home for the 8087 and for up to 256 kilobytes of additional memory.

The Hudson board is sold only by mail order and is offered in three configurations. Option 1 contains the 8087 support circuitry, assembled and tested, with empty sockets for memory and memory-support chips, for a price of $595. Option 2 contains 256 Kbytes of memory with empty sockets for the 8087 and its supporting chips, also for $595. Option 3 is a fully populated board with 256K of memory and the 8087, with everything tested, for $995.

A 21-page manual and a diskette containing demonstration programs and memory diagnostics are supplied with the board.

The Hudson board is very flexible. Its memory can have addresses beginning at any even multiple of 64 Kbytes above the

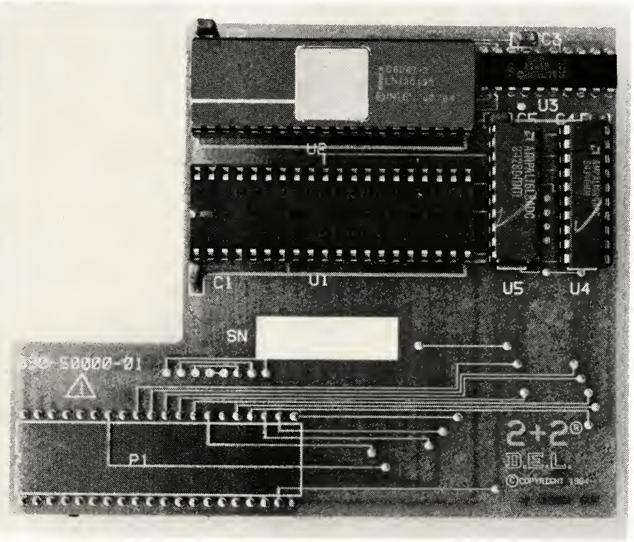


Photo courtesy of the author

Figure 2. The 2+2 board from D.E.L. Professional Systems.

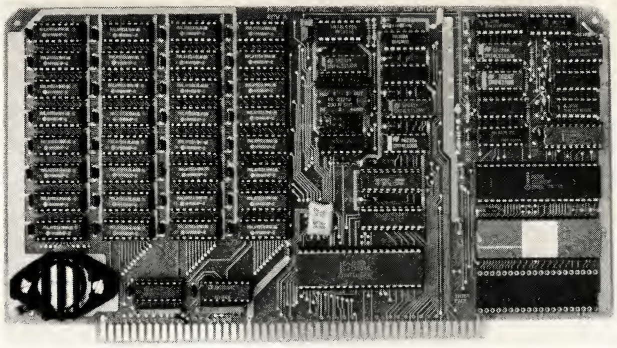


Photo courtesy of the author

Figure 3. The Z-Support Board from Hudson and Associates.

192K of random-access memory (RAM) on the '100's main board.

Instead of relying on a DIP switch setting, the address of the Hudson board's memory is established inside a programmable read-only memory (PROM) chip. The most common beginning addresses for additional memory are at 192K and 448K, corresponding to a fully populated main board and to a main board plus a 256K memory-expansion board, such as the Z-205 from Heath/Zenith. PROMs for these two memory configurations are supplied with the board.

There is further flexibility in the way the board handles the interrupt generated by the 8087 when it detects certain abnormal conditions, such as an attempt to divide by zero. A set of solder pads on the board allows this signal to be routed to any of the eight interrupt lines on the S-100 bus or to the non-maskable interrupt (NMI) on the 8088.

The 8088's NMI is the route assumed by the popular Microsoft FORTRAN and Pascal compilers; so, many users will wish to make that connection. That involves soldering a small jumper to the top of one of the card-edge connector pads, as shown in Figure 4. This must be done carefully to avoid getting solder on the gold contact surface.

The other two 8087 boards provide this connection to the NMI but make no provision for alternate use of the S-100 bus interrupts.

Another degree of flexibility is provided by two more sets of jumper pins controlling timing. One set allows the board to be optimized for RAM chips of different speeds; the other allows fine adjustment of the timing of memory-access commands sent out by the 8288 bus-controller chip. This makes it easier to "tweak" this board to work reliably at high clock rates.

Those who are adventurous and wish to save money may be tempted to buy Option 1 with the expectation that they can add the memory chips themselves; after all, 64-kilobit RAM chips have gotten very cheap lately.

I confess I did that, but I don't recommend it. You have to round up about ten other ICs, including an 8203 dynamic RAM controller which costs $40; as well, you must find a tapped delay line and program your own address-translation PROM. Unless you have access to such parts and to a PROM programmer, you had better buy the assembled and tested version. (The board is not designed to accept 256-kilobit RAM chips, by the way.)

Software support

OK, now you've got an 8087 in your machine, and you're eager to take it for a test drive. You load in that popular benchmark, the Sieve of Eratosthenes, run it with bated breath and stopwatch in hand—and nothing has changed.

The Sieve uses only integer and logical commands; the 8087 never saw anything it liked.

So you try that pokey BASIC graphics program that computes a bunch of sines and cosines. You know *that* uses lots of floating point.

Again, no change; Heath/Zenith BASIC doesn't know about 8087s, and all of the software floating-point emulation is deeply built into the language.

No, don't fire off an angry letter to Intel. You've got to go another step. You must find a program which supports the 8087, in that it uses the machine-language instructions which call the 8087 into action.

Does that mean *you* have to program in machine language? No, there are lots of compilers which offer 8087 support in a variety of programming languages. However, there are, as yet, only a few ready-made application programs which do so. Let's take a look at some of each.

New programs appear almost daily, so this will not be a complete review of programming support for the 8087. However, it will at least give you an idea of what kinds of things are available and what they might cost. Note that all prices mentioned are list prices; substantial discounts are frequently offered.

Let's take the compilers language by language.

BASIC

BASIC presents problems.

The reason, as mentioned above, is that Microsoft started off its 16-bit version of BASIC with a different floating-point format, not the IEEE format used by the 8087.

Thus Z-BASIC, GW-BASIC, and the Heath/Zenith BASIC Compiler (which is the Microsoft BASIC compiler, BASCOM, Version 5.4, adapted to the '100) all use a dialect foreign to the 8087. You can do a translation each time a floating-point operation is required, but this slows things down.

MicroWay offers a modified library for BASCOM which runs on the Z150 and which uses the 8087. It is called 87BASIC and lists at $140. You must send your copy of BASCOM to MicroWay. They perform the surgery needed to add support for the 8087 and return the altered compiler to you. A new version of BASCOM which uses the 8087 is rumored to be ready for release by Microsoft, but it may be a while before this is ported to the '100. These two products are machine-specific because graphics commands are built into them, and the graphics display commands are very different in the '100 and the '150.

It is worth mentioning that MicroWay is a specialty shop for 8087 users. It offers its own products, plus nearly all the available 8087 software from other firms, as well as the chips themselves, at discount prices. A similar shop is Hauppage Computer Works.

Softaid offers a BASIC compiler called MT-BASIC, with versions for the CP/M, PC-DOS, and MS-DOS operating systems. The latter two support the 8087 and cost $79.95. There is an interpreter mode for program development. Graphics on the H/Z100 are not supported.

BASIC interpreters are inherently slow, since they translate each line of code to machine language as your program runs. It would seem that a speed-up in the floating-point operations alone wouldn't help much. However, two firms offer BASIC incremental compilers

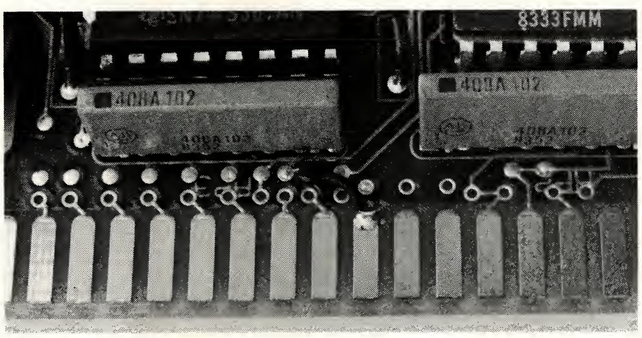


Photo courtesy of the author

Figure 4. A set of solder pads on the Hudson board allows routing an 8087 error-interrupt signal to any of eight interrupt lines on the S-100 bus or to the non-maskable interrupt on the 8088.

that make use of the 8087.

These products compile the program into machine-language instructions as you enter it, line by line; so, the program is ready to go when you type RUN. They are both written for the IBM PC; thus they should run on the H/Z150.

One of the two, Morgan Professional BASIC ($345), is closely linked to the IBM architecture. Better BASIC from Summit Software ($199 plus $99 for the 8087 extension) is modular, so that it can be adapted to various machines.

Pascal

For Pascal users there are at least two choices, one in several versions.

The popular Turbo Pascal, from Borland International, is available with 8087 support for a price of $89.95. It is a very good program-development system with a built-in editor and linker. Everyone who uses Turbo Pascal is impressed by the very fast compile times and by its ease of use.

Heath/Zenith sells Microsoft Pascal ($295).. This compiler comes with four separate math libraries. One of the math libraries has subroutines which automatically sense the presence of an 8087 and use it if it is there, but which will also run on a system that doesn't have one. Another library generates floating-point instructions rather than calls to subroutines, resulting in smaller code modules.

The other two Microsoft Pascal math libraries ignore the 8087 altogether. One does relatively fast limited-precision floating point in software. The other does *decimal* (base-ten) floating point.

Roughly the same compiler can also be obtained from Microsoft (Version 3.3, for $300); IBM Pascal Version 2 ($350) is also very similar.

FORTRAN

FORTRAN continues to be the workhorse of scientific and engineering computation (despite its bad image among students of computer science and devotees of "modern computer languages," who regard it as old-fashioned and inflexible). Thus it is not surprising to find a lot of 8087 support provided for this language.

Heath/Zenith sells Microsoft FORTRAN for $195. (This is currently Version 3.2, which is a nearly complete implementation of FORTRAN-77—including the COMPLEX data type, which was omitted in earlier versions.) This has the same four-library feature as Microsoft Pascal; in fact, the FORTRAN compiler is written in Pascal and uses many of the same modules as its sibling.

Microsoft will sell you Version 3.3 of this compiler for $350. Nearly the same compiler is offered as IBM FORTRAN Version 2.0 ($350); however, it does not allow the COMPLEX data type.

Three other FORTRAN compilers with 8087 support have recently become available; all three are complete implementations of FORTRAN-77.

Ryan-McFarland FORTRAN ($595—also sold as IBM Professional FORTRAN, same price) *requires* the 8087 and Version 2 or higher of MS-DOS. The IBM version will run on the H/Z100 if the MS-DOS linker is used instead of the IBM-supplied linker.

Digital Research's FORTRAN-77 will run without the 8087, but Lutowski, in the article mentioned above, reports that it is *very* slow without it. I have no first-hand experience with this compiler or with the third, from Lahey Computer Systems ($477). Because these are complete versions of the American National Standards Institute (ANSI) 1977 FORTRAN standard language, they make it easier to transport to your microcomputer programs originally written for larger machines.

SuperSoft has a FORTRAN compiler for $325 plus $50 for 8087 support. This compiler follows the FORTRAN-66 language definition, which means it lacks some important new features embodied in the FORTRAN-77 standard. Among the features it lacks are IF-THEN-ELSE logical structures, which make it easier to write programs you can understand when you look at them six months later. It does allow the COMPLEX data type.

C

There are a great many C compilers, and most of them support the 8087. For a detailed comparison of some of these, see "Nine C Compilers for the IBM PC," by Ralph Phraner, in the August 1983 *Byte*.

C Ware's DeSmet C ($159) is quite

References

There are many books which discuss the architecture of the IBM PC, and a few of these contain information about the 8087 and its use.

8087 Applications and Programming for the IBM PC and other PC's, by Richard Startz (Brady, 1983), is the text distributed by Heath with the Z-216. It contains a good description of the 8087 and of how it works with the 8088.

At the time this book was written, there was no compiler support for the 8087, so the programming examples are assembly-language subroutines which may be used with the Microsoft BASIC interpreter.

Startz is evidently an applied mathematician, and about half of the book is devoted to fairly high-powered numerical and statistical methods. An optional program disk is available from the publisher.

The "bible" on programming the 8086 family of microprocessors, including the 8087, is the Intel *iAPX 86/88, 186/188 User's Manual, Programmer's Reference* (Intel, 1983, order number 210911-001). Further useful dope from Intel is found in Application Note 113, *Getting Started with the Numeric Data Processor* (Intel, 1981, order number 207865-001).

The designers of the 8087 and the 8086, John F. Palmer and Stephen P. Morse, have written *The 8087 Primer* (John Wiley and Sons, 1984). This contains a good description of how the two processors work together. The differences between the 80287 and the 8087 are explained.

There is a long chapter on the 8087 in *Assembly Language Programming for the IBM Personal Computer*, by David J. Bradley (Prentice-Hall, 1984).

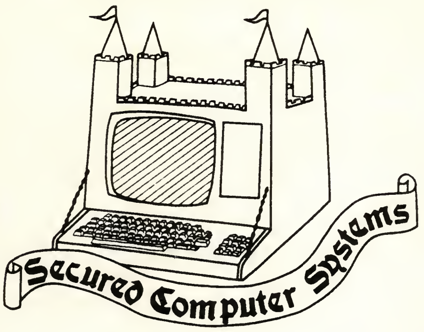
Bradley was a member of the original design team for the IBM PC, and he clearly knows his way around the hardware and software. Those of you with H/Z150s will find the book especially useful, but more than half of it pertains to the programming of any MS-DOS computer.

Each of the 8087's commands is thoroughly discussed. There are lots of programming examples, but all in assembly language, as the title suggests.

A couple of articles have discussed the adequacy of the compilers available for MS-DOS. Two are now somewhat out of date, but still worth reading.

In *Byte* for August, 1983, Ralph Phraner reviewed "Nine C compilers for the IBM PC." Chip Barnaby and Charlie Huizenga compared three FORTRAN compilers in *PC World* for February, 1984 ("But Is It Really FORTRAN?"). The new Microsoft C and the Ryan-McFarland FORTRAN compilers are described in articles in the *IBM Systems Journal*, Vol. 24, No. 1, 1985. (See "The C Programming Language and a C Compiler," by R. R. Ryan and H. Spiller of Microsoft, and "Design Considerations for IBM Personal Computer Professional FORTRAN, and Optimizing Compiler," by M. L. Roberts and P. D. Griffiths of Ryan-McFarland.)

Benchmarks appear frequently in product reviews, especially in *Byte*. Jim Gilbreath and Gary Gilbreath have compiled run times on the Sieve on many different combinations of software and hardware. (See "Eratosthenes Revisited: Once More through the Sieve," *Byte*, January, 1983.) Similar data for SAVAGE appear in *Dr. Dobb's Journal* for September 1983 and July 1984. (See Ray Duncan's column, "16-Bit Software Toolbox.")



HEATH/ZENITH 88, 89, 90 PERIPHERALS

16K RAM EXPANSION CARD

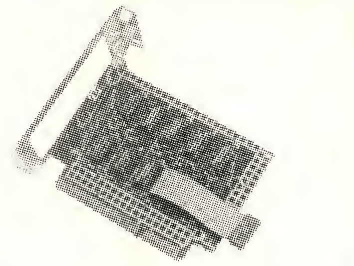
Expand your H/Z 88, 89 RAM Memory to a **FULL 64K** and begin using larger and more powerful programs with our **16K RAM** card.

Fully compatible with: Magnolia Microsystems and CDR CP/M and disk I/O interface cards.

Featuring: Complete installation instructions • Mounting bracket • 90 day warranty

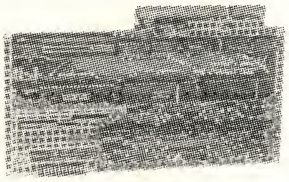
Field reliability record exceeding 3 years

Only $65.00
Shipping & Handling $5.00



2/3 PORT SERIAL CARD PORT PARALLEL I/O

". . . not your typical vanilla-flavored serial and parallel interface. . ."



Your H/Z 88, 89, 90 can now directly connect and operate EPSON, IDS, ANADEX, GEMINI, SILVER REED, NEC, SUPER 5, PROWRITER, OKIDATA, and many more line printers using CENTRONICS style parallel interface with our **2/3rds**, 2 port serial, 3 port parallel interface card. Or you may use all 24 digital lines in various configurations of input, output, or bidirectional modes for industrial control or data sampling.

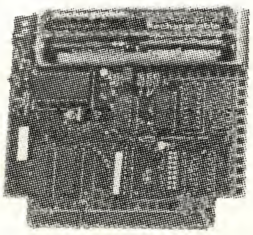
Features:
• 2 Serial Ports Supporting Ring Input, and External Clock • 3 Parallel Ports, 24 Total Digital Input/Output Lines • Fully Compatible with All Models of H/Z 88, 89, 90 using Heath/Zenith CP/M or HDOS • Now Supporting CP/M Version 2.2.04 • Choice of Centronics Line Printer Support Software for the CP/M or HDOS Operating Systems • Reduced Computer Bus Loading and Chip Independent Design

Complete with Installation Instructions, Documentation, 90 Day Warranty, Two Serial Cables and a Parallel Cable Internal to the Computer.

Price $199.00 **Second Operating System Driver $25.00**
Shipping & Handling $10.00

REAL TIME CLOCK

You will be able to perform time and date stamping for point of sales software, and bulletin board software or perform time studies as well as real time data sampling with our **REAL TIME CLOCK.** This peripheral card is a perfect companion to our 2/3rds card for industrial control and data sampling. STOP WATCH time study and alarm demonstration software is included for the CP/M operating systems. You will be able to view the current date and time on screen continuously or simply listen to an audible beep every fifteen minutes and the hour chimed or disable the clock entirely at your option.



Features:
• True I/O Addressing, Not Memory Mapped • User Selectable Address • Rechargeable Battery Backup Using Commonly Available Batteries • Installable on the Left or Right Side of the Computer (Left side operation requires our I/O expansion module.) • Month, Day, Year, Hour, Minutes, Seconds, 1/10's, 1/100's and 1/1000's of Second of Accuracy • Interrupt Capability Based on Tenths of Seconds, Seconds, Minutes, Hour, Day, Week or a Specific Date and time • CP/M Operating System Software Driver and Demonstration Programs • HDOS is not Supported.

Price $130.00 with Batteries **$114.00 without Batteries**
Shipping & Handling $5.00

HDOS is a reg. trademark of the Heath Co. CP/M is a reg. trademark of Digital Research
PRICES ARE LESS SHIPPING AND TAX IF RESIDENT OF CALIFORNIA
MAIL ORDER: 12011 ACLARE ST., CERRITOS, CA 90701 (213) 924-6741
TECHNICAL INFO/HELP: 8575 KNOTT AVE., SUITE D, BUENA PARK, CA 90620 (714) 952-3930

ZENITH data systems SERVICE CENTER

TERMS AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE — VISA AND MASTER CARD GLADLY ACCEPTED

Circle #157 on Reader Service Card

TaxAide

Income tax preparation worksheets
for use with your Lotus 1-2-3.

**Produces IRS Submittable Printouts—
No Special Forms to Buy**

* Easy to use—menu driven *
* Screen displays resemble actual IRS forms *

TaxAide Plus—Provides worksheets for Forms 1040, 1040A, 2106, 2119, 2441, 3468, 3800, 3903, 4562, 5695, and 6251; Schedules 1, A, B, C, D, E, F, G, R, SE, and W; plus IRS worksheets. Multiple Forms 2106 and 4562 and Schedules C and SE. TAXAIDE PLUS PRODUCES IRS SUBMITTABLE PRINTOUTS FOR ALL FORMS AND SCHEDULES, INCLUDING 1040 AND 1040A. Ideal for income tax preparers. **$59.95**

TaxAide Personal—Provides worksheets for Forms 1040, 1040A, 2106, 2441, and 5695; Schedules 1, A, B, D, G, R, and W, plus IRS worksheets. Two Forms 2106 are provided. PRODUCES IRS SUBMITTABLE PRINTOUTS (EXCEPT FORMS 1040 AND 1040A). **$29.95**

Requires only 192K of RAM—If you have more memory, you may still run your background program such as SideKick or Perks.

Requires Lotus 1-2-3 version 1A or later, and 360K capacity disk drives.

To order, send check to:

Software Applications of Wichita
2204 Winstead Circle
Wichita, KS 67226

Kansas residents add 4% sales tax.
For Visa or MasterCard orders, call (316) 684-0304.

TaxAide is a trademark of Software Applications of Wichita. Lotus and 1-2-3 are registered trademarks of Lotus Development Corp. SideKick is a trademark of Borland International, Inc. Perks is a trademark of Barry A. Watzman.

Circle #188 on Reader Service Card

ZIO-MATE-100 KIT

Analog-Digital/Digital-Analog/Digital TTL
Input/Output Board-Kit for
the Z-100 series computer

For *data* acquisition, *instrument* or *process control, robotics*, or any application involving analog/digital interface. Field proven for over six years as an integral component of our *LABMATE-V* system, now offered as a plug-in kit,
because *you asked for it!!!*

- 8 multiplexed single ended analog inputs, each with programmable gain and 12 bit resolution.
- 2 latched analog outputs-10 bit resolution.
- 24 TTL digital inputs
- 16 TTL digital outputs

The kit contains everything you need to convert your Z-100 into a data acquisition/control system: the ZIO board, cables, rear panel hardware and connectors, plus software and instruction manual.

Call or write for more details
or send your order with $975.00 to:

SPECTRO PRODUCTS INC.
385 STATE STREET, NORTH HAVEN, CONNECTICUT 06473
(203) 281-0122

Circle #174 on Reader Service Card

popular, but it handles only the 8088's small-memory model, in which a program's code and data regions are each limited to 64 Kbytes. In the DeSmet package, you also get an assembler and a debugger.

Lattice C ($500) has library functions which automatically make use of an 8087 if it is present; a similar feature is offered with Eco-C by Ecosoft ($49.95).

If you buy the Lattice compiler, make sure you get the current version, presently 2.15. Previous versions had a math library which was unbelievably slow, even with an 8087.

Microsoft has recently brought out a new C compiler (Version 3.0, $500), adapted from the one in its Xenix operating system; previous C compilers sold by Microsoft were versions of the Lattice compiler.

Other possibilities include Manx's Aztec C86—$199 for a basic version which does not use the 8087; $299 for the developer's system which does and also includes several program development aids.

C-Systems' C compiler ($199) also includes an assembler and debugger. Mark Williams C ($500) is one of a group of related compilers and cross-compilers which operate on a variety of different computer systems. Wizard C from Wizard Systems Software ($450) permits you to include in-line assembly-language code in C programs.

There are several incremental compilers for C. These offer easy program development and debugging together with fairly efficient machine-language output code. However, the code they generate cannot be as efficient as that produced by an optimizing compiler. Instant-C from Rational Systems costs $495, and Run-C from Lifeboat Associates costs $150. C-Terp, from Gimpel Software ($300), requires IBM-PC compatibility, so it should run on the '150 but not the '100.

Forth

Forth is a language with many fervent advocates (and critics). Its nature is such that it is easy to extend the language's vocabulary. Versions with floating-point arithmetic using the 8087 are offered by Harvard Softworks ($250), Laboratory Microsystems ($100 plus $100 more for the 8087 package), Mountain View Press ($150 + $85), and WL Computer Systems ($100).

Others

There are, of course, many other programming languages. Because of the immense popularity of the IBM PC, many implementations are specific to that machine, although they should also run on the H/Z150.

Generic MS-DOS versions which will also run on the H/Z100 are less common. I have noticed mention of 8087 support in advertisements for MS-DOS versions of Snobol from Catspaw ($95) and of Modula-2 from Logitech Systems ($495).

The number of ready-made applications programs which make use of the 8087 is still fairly small, although I expect it will grow rapidly now that the basic programming tools, efficient compilers, have been developed.

Programs which do extensive transformation of graphical images, such as AutoCAD, by Autodesk ($1,000 to $2,000, depending on options), benefit greatly from faster floating-point operations.

At least one spreadsheet program, SuperCalc 3, from Sorcim, uses the 8087; but this program requires close IBM compatibility. It is sold by Heath/Zenith for use on the '150 as SC-5063-2, $395. The earlier version available for the '100, SC-463-1, $195, does not use the 8087.

Libraries of mathematical and statistical functions which may be linked with programs written in FORTRAN, Pascal, and C are available from a number of vendors. Have a look at the ads in *Byte* or *PC Tech Journal* for the latest offerings.

The Microsoft assembler, MASM Version 1.27, is sold by Heath/Zenith as part of the Programmer's Utility Pack ($195). This version includes the full set of 8087 instructions, whereas the earlier version distributed as a part of Z-DOS did not.

MS-DOS's debugger DEBUG offers only partial support for the 8087. A more comprehensive floating-point debugger, 87DEBUG, is offered by MicroWay ($150), but it will not run on the '100.

A later version of the assembler is available from Microsoft (Version 3.0, $150) and includes a new symbolic debug program, SYMDEB. It also includes MAKE, a program to manage development of complex software systems, patterned after a Unix program of the same name. (Well, it's almost the same—under Unix, it's called "make", since there's a very strong lower-case fetish in that kingdom. By the way, Unix is a trademark of AT&T Bell Labs, and they want you to *remember* that.)

```
$TITLE: 'Sieve of Eratosthenes'
$PAGESIZE: 60
$NOFLOATCALLS
C
C  Counts all primes .LT. 2**14
C
         INTEGER*2 C,I,K,M,P
         INTEGER*4 IT
         REAL*4 SECS
         PARAMETER (L=8190)
         LOGICAL*2 F(L)
C
         IT = 0
         WRITE (*,*) '    Start'
         IT = ITIME(IT)
         DO 30 M = 1,10
           C = 0
           DO 10 I = 1,L
   10        F(I) = .FALSE.
           DO 20 I = 1,L
             IF (F(I)) GO TO 20
             P = 2*I + 3
             K = I + P
   15        IF (K.LE.L) THEN
   17          F(K) = .TRUE.
               K = K + P
               GO TO 15
             ELSE
               C = C + 1
             ENDIF
   20        CONTINUE
   30      CONTINUE
         WRITE(*,1000) C
         SECS = FLOAT(ITIME(IT))/100.
         WRITE(*,1001) SECS
         STOP
C
 1000    FORMAT(/'     End with ',I7,'  primes'/)
 1001    FORMAT('     Run time = ',F6.2,' seconds'///)
C
         END
```

Listing 1. The Sieve of Eratosthenes is used to test a computer system's efficiency in integer and logical operations.

Benchmarks

A benchmark is a short program which may be used to test the relative speeds of

computers having different hardware or software. A benchmark may be written to emphasize some particular type of operation; or one may be written to use a variety of different features in order to obtain a best estimate of performance on an "average" program.

To show how several available languages benefit from the addition of an 8087 to the H/Z100, and to compare their efficiencies on different types of problems, I have chosen four benchmarks.

The first of these, Listing 1, is the well known Sieve of Eratosthenes, which has been run on almost every known computer from a Cray to an abacus! (See "Eratosthenes Revisited: Once More Through the Sieve," by Jim Gilbreath and Gary Gilbreath, in the January 1983 *Byte*.) The Sieve makes *no* use of floating point arithmetic; it tests the compiler's efficiency on integer and logical operations.

INTSUM, Listing 2, sums the first 100,000 integers, expressed as double-precision floating-point numbers (multiplying by 2.0, dividing by 0.5, and dividing by 4 as it goes). Thus INTSUM measures the speed of pure floating-

Sources Mentioned

Autodesk, Inc.
150 Shoreline Highway, Building B
Mill Valley, CA 94941
415/332-2344

Borland International
4113 Scotts Valley Drive
Scotts Valley, CA 95066
408/438-8400

Catspaw, Inc.
P.O. Box 1123
Salida, CO 81201
303/539-3884

C-Systems
P.O. Box 3253
Fullerton, CA 92634
714/637-5362

C Ware
P.O. Box C
Sunnyvale, CA 94087
408/720-9696

D.E.L. Professional Systems (now called Gemini Technologies, Inc.)
12-11151 Horseshoe Way
Richmond, B.C., Canada V7A 4S5
800/663-8331

Digital Research, Inc.
P.O. Box DRI
Monterey, CA 93942
408/649-3896

Ecosoft, Inc.
6413 North College Avenue
Indianapolis, IN 46220
317/255-6476

Gimpel Software
3207 Hogarth Lane
Collegeville, PA 19426
215/584-4261

Harvard Softworks
P.O. Box 339
Harvard, MA 01451
617/456-3021

Hauppage Computer Works
358 Veterans Memorial Highway, Suite MSI
Commack, NY 11725
516/360-3827

Heath Company
Benton Harbor, MI 49022
Orders: 800/253-0570, 616/982-3411 from Michigan and Alaska;
information: 616/982-3285

Hudson and Associates
P.O. Box 2957
Santa Clara, CA 95055
408/554-1316

Intel Corp.
3065 Bowers Avenue
Santa Clara, CA 95051
408/987-8080

IBM Entry Systems Division
P.O. Box 1328
Boca Raton, FL 33432
305/998-2000

Laboratory Microsystems, Inc.
P.O. Box 10430
Marina del Rey, CA 90295
213/306-7412

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive, West
Suite 243
Rancho Palos Verdes, CA 90274
213/541-1200

Lattice, Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
312/858-7950

Lifeboat Associates
1651 Third Avenue
New York, NY 10128
800/847-7078

Logitech
805 Veterans Boulevard
Redwood City, CA 94063
415/365-9852

Manx Software Systems
P.O. Box 55
Shrewsbury, NJ 07701
800/221-0440

Mark Williams Co.
1430 West Wrightwood Avenue
Chicago, IL 60614
312/472-6659

Microsoft Corp.
10700 Northup Way
Box 97200
Belleview, WA 98009
206/828-8088

MicroWay Inc.
P.O. Box 79
Kingston, MA 02364
617/746-7341

Morgan Computing Co. Inc.
10400 North Central Expressway
Suite 210
Dallas, TX 75231
214/739-5895

Mountain View Press
P.O. Box 4656
Mountain View, CA 94040
415/961-4103

Rational Systems, Inc.
P.O. Box 480
Natick, MA 07160
617/653-6194

Ryan-McFarland Corp.
609 Deep Valley Drive
Rolling Hills Estates, CA 90274
213/541-4828

Softaid, Inc.
P.O. Box 2412
Columbia, MD 21045
301/792-8096

Supersoft, Inc.
P.O. Box 1628
Champaign, IL 61820
800/762-6629

Summit Software Technology, Inc.
P.O. Box 99, Babson Park
Wellesley, MA 02157
800/225-5800

Wizard Systems Software, Inc.
11 Willow Court
Arlington, MA 02174
617/641-2379

WL Computer Systems
1910 Newman Road
West Lafayette, IN 47906
317/743-8484

point arithmetic.

SAVAGE, Listing 3, has also been run on a very wide variety of computers. It consists mainly of double-precision function calls, so it tests the efficiency of the compiler's library routines for these functions. It also tests the accuracy of these functions, but I won't go into that aspect.

FOUR, Listing 4, computes 1 the hard way, by summing 1,000 terms of the Fourier series for a square wave, which would have a value of exactly one if the number of terms were infinite. It does this for ten values of the argument between 0.1 and 1.0; the sums are seen to oscillate about the nominal value of 1.0. This is the correct behavior for the finite series. Indexing is used to simulate a more realistic problem, such as a fast

```
$TITLE: 'INTSUM - Floating point math test'
$PAGESIZE: 60
$NOFLOATCALLS
C
        REAL*4 SECS
        REAL*8 A,B,C,D
        INTEGER*4 MAX,IT,I
        PARAMETER (MAX=100000)
C
        A = 0.DO
        B = 0.DO
        C = 2.0DO
        D = 0.5DO
        IT = 0
        WRITE(*,*)'     Start'
        IT = ITIME(IT)
        DO 10 I = 1,MAX
                A = A + I*C
   10           B = B + I/D
        A = (A + B)/4.0DO
        WRITE(*,110) A
        SECS = FLOAT(ITIME(IT))/100.
        WRITE(*,100)SECS
        STOP
C
  100 FORMAT('      Run time = ',F6.2,' seconds'///)
  110 FORMAT(/5X,D30.20/)
        END
```

Listing 2. INTSUM sums the first 100,000 integers, multiplying by 2, dividing by .5, and dividing by 4 as it goes. It is used to test the speed of pure floating-point arithmetic.

```
$TITLE: 'SAVAGE'
$PAGESIZE: 60
$NOFLOATCALLS
C
        REAL*4 SECS
        REAL*8 A
        INTEGER*2 I
        INTEGER*4 IT
C
        IT = 0
        WRITE(*,*)'     Start'
        IT = ITIME(IT)
        A = 1.0DO
        DO 10 I = 1,2499
   10     A = DTAN(DATAN(DEXP(DLOG(DSQRT(A**2))))) + 1.0DO
        WRITE(*,100)A
        SECS = ITIME(IT)/100.
        WRITE(*,101)SECS
        STOP
  100   FORMAT(/5X,F20.10/)
  101   FORMAT('      Run time = ',F6.2,' seconds'///)
        END
```

Listing 3. SAVAGE consists mainly of double-precision function calls; it tests the efficiency of a compiler's library routines for these functions.

Fourier transform. Sine functions and single-precision arithmetic are used in FOUR—except with Turbo Pascal, as noted below.

Table 1 gives run times for these four programs in interpreted and compiled BASIC and in two versions each of Pascal, FORTRAN, and C.

In Table 1, the column headed 8087 is marked Y for runs made with an 8087, N for runs without one. Under MS-FORTRAN, there is a third entry marked A. This is for the ALTMATH library, which offers faster run times than the 8087 emulator library but which ignores the 8087. A similar option is available under MS-Pascal.

Each version of Turbo Pascal offers only one degree of floating-point precision. In the version with software floating point, each number occupies six bytes and offers a precision intermediate between conventional single precision (four-byte) and double precision (eight-byte). The 8087 version employs eight-byte floating-point numbers.

It is immediately apparent that the 8087 speeds up floating-point calculations by a very large factor—10 to 30 in most cases. It is also clear that the compilers tested differ quite a bit in their efficiency.

Overall, the poorest is DeSmet C, and the best is Ryan-McFarland FORTRAN, but the spread in run times on the floating-point benchmarks is only about 4:1 at worst.

The first two rows of Table 1 also show the huge gain in speed which you get by going from interpreted BASIC to compiled BASIC. The present version of the BASIC compiler can't use the 8087, but it produces code which is about as good as any of the other compilers when they are forced to do floating-point calculations in software.

The MS-FORTRAN and MS-Pascal programs were run using the $NOFLOATCALLS option. (In Pascal, it's $floatcalls–.) This option gives the fastest run times.

The Lattice C programs were run in both the small- and large-memory models. Times in Table 1 are for the small model; the increase in run times when the large model is employed is only 5% to 10%.

The computers used were a Z110 with an 8087 and a Z120 without one. All programs except those using Ryan-McFarland FORTRAN were run under Z-DOS Version 1.25.

If your microcomputing includes any substantial portion of work with floating-point numbers, you will surely want to make use of the marked enhancement in power you can get by installing an 8087.

The prices of these little marvels are falling rapidly, and the amount of support for them is rising even more rapidly. Give one a try—you'll be pleased with the result. △

```
$TITLE: 'FOUR - Fourier series with indexing'
$PAGESIZE:60
$NOFLOATCALLS
C
       REAL*4 ARG,PI,SUM(1000)
       INTEGER*2 ML,I,J,N
       INTEGER*4 IT,ITIME
       PARAMETER (ML=1999)
       PARAMETER (PI=3.1415926)
C
       IT = 0
       WRITE(*,*)'     Start'
       IT = ITIME(IT)
       DO 20 J = 1,10
         ARG = J/10.
         SUM(1) = SIN(ARG)
         DO 10 I = 3,ML,2
           N = (I+1)/2
   10      SUM(N) = SUM(N-1) + SIN(I*ARG)/I
         SUM(N) = 4.*SUM(N)/PI
         WRITE(*,1000)ARG,SUM(N)
   20    CONTINUE
       SECS = FLOAT(ITIME(IT))/100.
       WRITE(*,1001)SECS
       STOP
C
 1000  FORMAT(10X,F5.1,F15.5)
 1001  FORMAT(/'      Run time = ',F6.2,' seconds'///)
C
       END
```

Listing 4. FOUR sums 1,000 terms of the Fourier series for a square wave. This would produce a value of exactly one if the number of terms were infinite.

Compiler (Version)	8087	SIEVE	INTSUM	SAVAGE	FOUR
Z-BASIC	N	2173	1160	145*	306
BASCOM (5.40)	N	14.6	346	53*	79
MS-FORTRAN (3.20)	N	13.0	516	249	369
	A‡	13.0	253	161	75.1
	Y	13.0	27.1	6.8	11.7
R-M FORTRAN (1.0)	Y	10.3	29.6	4.0	8.2
MS-Pascal (3.20)	N	10.4	572	241	354
	Y	10.4	28.4	6.3	10.9
Turbo Pascal (3.0)	N	13.3	304ˣ	189ˣ	183ˣ
	Y	13.3	108	7.9	13.9†
DeSmet C (2.4)	N	10.8	435	497	347
	Y	10.8	50	12.6	43
Lattice C (2.15)	N	10.1	813	476	550
	Y	10.1	104	6.8	21.9

* Single precision
ˣ Intermediate precision
† Double precision
‡ These runtimes were produced using the ALTMATH library provided with MS-FORTRAN.

Table 1. The four benchmark programs given in Listing 1-4 were run using a number of popular language packages. The run times for each are shown here in seconds. A Y in the column headed "8087" indicates the run was made with the 8087; an N, without. Under MS-FORTRAN, the A in this column indicates the ALTMATH library was used.

The Eight-Bit World

Walter J. Janowski

We've all noted and enjoyed the addition to *Sextant* of the "Z100 Notebook" column by Alison Phillips; wouldn't it be nice if there were a similar offering to cover Heath/Zenith's eight-bit world?

I hope this column will do just that. There are an awful lot of H8s and H89s out there, and contrary to popular opinion, activity involving these machines is continuing.

Hardware obsolescence

During my brief stint as a salesman of computer products (one of those "other" brands, unfortunately), one of the most asked questions from prospective buyers was: "If I buy this unit, will it become obsolete?" I think it's time we cleared up some popular misconceptions about that term.

The question is usually asked to determine if the current model will soon be replaced by a fancy new one with all the bells and whistles that any corporate executive with a twelve-year-old's fascination for bright lights and flashing colors could ask for. The Random House dictionary gives several definitions of obsolete, one of which is "of a discarded or outmoded type; out of date."

That seems to cover the nervous buyer's question, but a more concrete definition is "no longer in general use; fallen into disuse." And therein lies the difference.

If a machine functions properly and still performs the tasks for which it was acquired, can it ever be obsolete? This article is being written on my faithful old H89, using the PIE editor from The Software Toolworks, running under (gasp) the Heath Disk Operating System (HDOS).

Sure, I can't run Lotus 1-2-3 or create bit-mapped color graphics. But then, in all honesty, I can't think of any reason why I'd ever want to. I have just about all the word processors, spreadsheets, languages, utilities, games, etc. that I could ever want. (And they all fit into 64K, too!)

If my system still serves all my purposes, how can it be obsolete?

HDOS and beyond

Way back when, in the good old days of home computing, any jokers who turned out a system with a disk drive also produced their own version of the disk operating system to end all disk operating systems. One by one, we watched as most of these systems (and many of the companies, for that matter) faded off into the sunset. Most of the machines based on the Z80 and 8080 processors adopted the CP/M operating system as their de facto standard.

One curious and stubborn little holdout was Heath's own HDOS.

The pros and cons of the HDOS vs. CP/M debate have been well voiced over the years, and I will not attempt to resolve the conflict here. But HDOS has some very good points that have helped add to its longevity.

First and probably most important, Heath was one of the very few companies (if not the only one) that made the entire source code for its operating system available. This was coupled with the fact that the user base that naturally grew around a kit computer was very much hobbyist oriented. Those two facts enabled HDOS users to understand, dissect, modify, and generally fool with their operating system in a way barred to users of other equipment.

As the song goes, to know it is to love it, and HDOS maintains a loyal and passionate following to this day.

This column will deal with HDOS as well as CP/M on a regular basis. Again, many HDOS users now feel left out in the cold more than ever. I will try to keep up on any new developments involving HDOS (including the often-mentioned HDOS 3) as much as possible. Also, this column can serve as a point to bring together in one place the many odds and ends of information on HDOS that have come up in the past.

Super HDOS?

If you are still using HDOS, there is no excuse for your not having Super Sysmod2 from Jim Teixeira's SoftShop.

Super Sysmod2 is a system enhancement for HDOS; it creates a modified version of the SYSCMD.SYS, PIP.ABS, and SYSHELP.DOC files normally distributed with HDOS version 2. If you found some areas of HDOS inconvenient or not quite as powerful or flexible as you liked, you may find many of your complaints remedied by Sysmod2.

First of all, Super Sysmod2 allows you to abbreviate most HDOS commands to one or two letters. This is a big plus if you're a lazy typist like myself.

If you use both HDOS and CP/M and find the transition from one-character drive names like A: to HDOS's SY0: too tough to handle, Super Sysmod2 allows you to specify your drives with a single digit (e.g., SY1: becomes 1:). Thus, MOUNT SY0: becomes M0; CAT SY1: becomes C1.

And that's not all. Tired of text or long directories scrolling off the screen before you can read them? By typing a period (.) before any command, screen output will scroll 24 lines and stop, waiting for you to strike the little-utilized SCROLL key on your keyboard.

Want to create text files that will execute like SUBMIT files in CP/M or .BAT files in the Microsoft Disk Operating System (MS-DOS) ? The DOCOM command will create .ABS files which do just that.

Recently, I was fretting over the fact that I had no HDOS file-manager programs comparable to the public domain programs SWEEP or VFILER for CP/M. Then I remembered I didn't need one because Super Sysmod2 had one already built in.

From the HDOS command level, the FM command will allow you to scan a diskette, file by file, and copy, delete, view, rename your files, and more. You can even alter the file-attribute flags (read-only, etc.) including the previously unchangeable "L" (locked) flag.

That's a pretty good collection of utilities already, but it doesn't stop there. You can get an alphabetized directory, change the label on a disk, create a new HDOS prompt, copy files by date, print files from the command line, send a directory listing to the printer, copy corrupt files, and more.

And the best part is that it's only $29.95!

So if those old HDOS disks are buried away in the back of your file box, dig them out and dust them off. Then get a copy of Super Sysmod2 and find out how easy it is to enjoy using HDOS even more.

Public domain software for the H8/H89

While concerning ourselves with the computer side of the world, we sometimes forget just how very many H/Z19 terminals have been sold by Heath/Zenith. The '19 was for many years one of the best selling terminals on the market.

Consequently, there are a lot more computers feeding the '19 terminals than just the H8 and '89. A little careful poking around in the public domain world turns up quite a few programs written to take advantage of the '19's terminal functions and graphics capabilities.

My own favorite and an indispensible addition to any CP/M system is VFILER.H89. This is the popular VFILER program, similar to SWEEP and WASH; the .H89 version not only gives you total control over your files, but does it in a convenient way only made possible by acknowledging the capabilities of the H/Z19.

Many similar utilities present your files one at a time, scrolling off the screen as lines progress; but VFILER presents your entire disk directory in a five-column format with each file individually

HDOS users don't need to feel left out either.

accessible via WordStar-like control functions.

For those not familiar with these file-handling utilities, imagine PIP and STAT replaced by one handier, simpler, infinitely more flexible program. You are allowed batch copying, batch delete, file and disk status, user-area transfers, and a whole lot more.

If you don't have a copy of VFILER.H89 yet, get your hands on one as soon as you can.

For you more serious CP/M programmers, there's HSTAT. It's a simple little program which, when invoked, provides a complete status listing of your system hardware, with the display all slicked up with '19 graphics. HSTAT will provide memory locations of CP/M's console command processor (CCP), basic disk operating system (BDOS), and basic input/output system (BIOS). It will also give you TPA size, drive and I/O port status, the number of soft errors since the last cold boot, and more. Even if you never need this level of internal information, it's still pretty neat to have on hand.

HDOS users don't need to feel left out either.

If you're tired of wrestling with the type of display you get from the CATalog command (even if you have Super Sysmod), you might be interested to know there is a program similar to the many sorted directory programs of CP/M.

SD.ABS by John Stetson gives a sorted HDOS directory in a format consistent with the CP/M versions. However, not

only does it use H/Z19 terminal capabilities, it provides many more features. You can get long or brief directory listings, sorted or unsorted listings, drive selection, and direct output to a list device. In general, any of the commands allowed by SD.ABS may be combined into a single command line. For example: SD 2/D *.A?M would provide a long output sorted by date for files on drive SY2: with a wildcard extension of .A?M.

The best way to track down these and other programs is to browse your local RCP/M bulletin boards and also HUG SIG, CompuServe's special interest group for the national Heath Users' Group. Most Heath/Zenith-specific programs have some mention of the H8, H/Z89, or H/Z19 in the program name.

For those of you not fortunate enough to have access to a modem, or wealthy enough to deal with long-distance phone rates (even with MCI), I will be glad to provide copies of any public domain software mentioned in any of my columns to anyone who sends me a blank diskette and $2 to cover postage and handling. I can handle 5¼″ formats, hard- and soft-sectored, single- and double-sided, HDOS and CP/M.

Help!

With all the action centered around Heath/Zenith's computers with the three-digit model numbers, it often seems as if the eight-bit users have crawled off to hide in the woodwork. If you would like to have this column continue to provide information on the state of the Heath/Zenith eight-bit world, you can help me to do so.

If you have questions, problems, information or software you've discovered, ideas, suggestions, or gripes, drop me a line and let me know.

Vendors, if you have new products you might like mentioned, old products you might like remembered, or ideas you might like to present to the eight-bit community, let me know.

For all correspondence, if you would like a personal reply, please include a self-addressed stamped envelope with your letter.

I can also be reached via CompuServe and the HUG SIG (I.D. number 72376,1652).

With your help, we can keep our so-called "obsolete" computers running and happy for a long time to come.

Walter J. Janowski
1505 East Evans Avenue
Valparaiso, IN 46383

Ordering Information

Super Sysmod2, $29.95.
SoftShop
35 Shadow Oak Drive
Sudbury, MA 01776
617/443-9693

△

INTERROGATOR™

Drive Diagnostic Program for the ZENITH Z/H-100 Series

Now available for H/Z148, H/Z150–160, Z171, Z200

Diagnosing disc drive problems has just become a great deal easier.

Interrogator,™ Dysan's economical new drive diagnostics package, can run a drive through its pace in minutes. To tell you exactly what's wrong. Or right.

Just like that.

Test results come right up on the screen of the machine you're testing. Or you can print out hard copy records.

The Interrogator package includes a Digital Diagnostic Diskette,™ complete test instructions and software programmed to perform a whole battery of sophisticated tests.

Things like head radial alignment. Head positioner linearity. Head azimuth alignment. Index timing. Spindle speed. Read/write verification. And over a dozen others.

Interrogator is here now for the Zenith Z/H 100 Series. Testing both 5¼″ and 8″ models. Versions for other computer models are on the way.

Whether you're going for one drive or need to assure the compatibility of a company full of drives, Interrogator is the perfect tool.

And it's just the kind of thing you'd expect from Dysan. The leader in quality, innovative media and diagnostic tools.

Interrogator. What could be easier?

Just this. A toll free phone call to get more information. Or place an order. (800) 551-9000.

Dysan®

5100 Patrick Henry Drive
P.O. Box 58053
Santa Clara, CA 95050
(408) 988–3472

Circle #184 on Reader Service Card

How to Use PROMPT on the '100 and '150

Displaying the current drive letter is only one of many things MS-DOS's PROMPT command can do for you.

William M. Adney

The powerful PROMPT command is probably the most overlooked feature of the latest versions of the Microsoft Disk Operating System (MS-DOS) and its derivatives (e.g., PC-DOS).

PROMPT allows you to do some fancy things—such as changing the prompt display colors on a monitor, adding a time and date display to the prompt, or silencing the keyclick on the H/Z100 computer.

But the best part is that you don't have to be an assembly-language programmer or a technical expert to take advantage of these features. You can do these things with a modest knowledge of some of the control and ESCape sequences that control features of your hardware.

The only requirement is that you must have MS-DOS (or PC-DOS) version 2 or later. (None of the version 1 implementations, including Heath/Zenith's Zenith Disk Operating System, included the PROMPT command.)

If you spend an hour or so reading this article and trying the examples on your computer, you may be absolutely stunned at some of the things that you can do. You will also be able to impress your family and friends with a customized prompt that is unique to *your* system. This article provides the "how to," complete with practical examples, as well as some of the theory on the way the command works.

The examples presented in this article will work with both the H/Z100 and the H/Z150 series. Where differences exist, I have shown equivalent examples for both the '100 and the '150 where possible. Examples shown for the '150 should work for all IBM Personal Computers and hardware-compatibles.

William M. Adney is a member of the faculty at the University of Texas at Arlington, as well as a freelance writer and computer systems consultant. He wrote the FlipFast *guides to CP/M and MS-DOS.*

What is the PROMPT command?

The MS-DOS prompt display is usually in the form of A>. In that basic case, the current-drive designator is A followed by a separator (the greater-than sign—>) and the cursor. The PROMPT command allows you to change that display.

Let's say that you prefer a colon following the drive designator instead of the greater-than sign. Just use the command:

PROMPT $N:

Here, the $N is PROMPT's standard specification to indicate the current drive; followed by your colon, the prompt display is changed to the form of A:.

To return to the original form of the MS-DOS prompt, simply enter PROMPT. No matter how you change the actual display with the command, the PROMPT command by itself will always return you to the standard MS-DOS prompt.

This command is commonly referred to as a system, built-in, internal, or system-resident command. This means it is contained in (is built into) the MS-DOS command interpreter (COMMAND.COM). COMMAND.COM is loaded into memory upon bootup; so, you can execute the PROMPT command regardless of the drive indicated by the command prompt, and regardless of what disk is in the drive.

The general form of the PROMPT command

The PROMPT command has the two general forms that I have already indicated. The PROMPT command by itself simply returns you to the standard MS-DOS prompt. When the command is followed by special prompt parameters, you can easily change the display to suit your individual needs.

Table 1 lists the PROMPT command's possible parameters and their effects.

Although the prompt parameters themselves may be input in upper- or lowercase, all other characters are interpreted *exactly* as input. This is particularly important because the $E allows you to incorporate ESCape sequences into your prompt.

For example, you can clear the screen with an ESCape E on the H/Z100, or with an ESCape [2J on the H/Z150. But ESCape e and ESCape [2j (note the lowercase letters) will not clear the screen, because they are not recognized by the hardware.

I have used the uppercase for the PROMPT parameters in all examples since they make the examples more readable. Any time I give a character in lowercase, that indicates it must be typed that way.

All other characters not shown in Table 1 may also be used in the command, except that they perform no function. (These characters, such as $A, are useful in some situations and are called null characters. I will take a look at one use of the null characters in a later example.)

One other restriction in the use of the PROMPT command (or any MS-DOS command) is that it can be no longer than 128 characters.

PROMPT has six letters; it must be followed by a space, for a total of seven characters. In addition, all commands are followed by a CTRL-M code, which is generated when you press the RETURN key. That is a total of eight characters required for the command. This leaves a maximum of 120 characters which can be used for the prompt parameters themselves. (Similarly, the COPY command allows a total of 122 characters for file names and other parameters.)

There is a display problem when the command line is longer than 80 characters; you must use a "line feed" sequence on your terminal in order to extend the command sequence to the next line without sending the command to the system. Virtually all microcomputers and their operating systems allow you to enter a CTRL-J sequence as a line feed. The H/Z100, however, also allows you just to press the LINE FEED key. On the H/Z150, a CTRL-RETURN can be used.

When you have finished entering the

complete command line, just press RETURN as usual to send the command to the system.

Getting started with PROMPT

I have found that the easiest way to experiment with a new command is to set it up in a batch file. That makes it easy to edit a complicated command, and you can save various versions of the command until you find one that you really like. I usually choose a simple file name, like P.BAT, and later versions can be called P1.BAT, P2.BAT, and so on.

By the way, one characteristic of a batch file is that it must contain only standard ASCII characters. You must use an editor that generates a standard ASCII file, such as MS-DOS's EDLIN or the non-document mode of WordStar.

Many word processors, such as Microsoft Word and MultiMate, do not easily generate standard ASCII files—they include special characters which are used by the word processing software. EDLIN is a perfect editor for adding and changing small files since it is supplied as part of all standard MS-DOS operating systems.

Many of you know that most operating systems include a "free" editor of one kind or another. And some people believe that those editors are not even worth that price. It is difficult to imagine an editor that is harder to use than CP/M's ED. MS-DOS's editor, EDLIN, however, is much easier to use.

The fact that EDLIN can program function keys to perform various functions (such as copy one character or copy all characters) provides a useful way to create small files. Batch files, configuration files, and other small files can be quickly created and/or modified with EDLIN.

(However, I do not recommend that you plan to use EDLIN to create letters or to do large programming tasks—a full-screen editor or word processor like WatchWord or WordPerfect is much more suited to the task.)

Which subdirectory am I in?

One of the best features of MS-DOS version 2 is that you can create subdirectories for special purposes. Although this is a nice feature which allows you to organize various programs and data, you can easily forget which subdirectory you are in. Wouldn't it be nice if the subdirectory could be displayed as part of the command prompt? It is very easy to do that with the PROMPT command—as shown in Figure 1.

If you do not like my two equal signs before the prompt symbol, simply eliminate both of the $Q parameters.

Note that I have shown a space between the drive/directory parameter ($P) and the first equal sign ($Q) so that I will have a space between the directory name and the equal sign. This example demonstrates the importance of spaces as well as

PROMPT (Return to default prompt)

PROMPT <parameters>

Parameters	Description
$B	Bar (¦) symbol
$D	Date display for current system date
$E	Escape character (1B hex)
$G	Greater than (>) symbol
$H	Backspace and erase previous character (like CTRL-H)
$L	Less than (<) symbol
$N	Current drive letter
$P	Current drive letter and directory name
$Q	Equal sign displayed within prompt
$S	Space displayed as first character (leading space)
$T	Time display for current system time
$V	Version number of the DOS (like the VER command)
$_	Underline character to get a CR-LF sequence
$=	Equal sign to set values for prompt
$c	Null character; c represents any character not listed above

Note: Except for the underline symbol and equal sign, the characters shown may also be input in lower case. The null character, indicated by $c, is any upper- or lowercase character not listed above.

Table 1. The parameters of the PROMPT command. (The general forms of the PROMPT command are shown at the top.) Except for the underline symbol and equal sign, the characters shown following the dollar sign may be input in either upper- or lowercase. For the null character, "c" may be any upper- or lowercase character not listed above.

providing useful information on subdirectory names.

Adding time, date, and a message

Since that was easy, let's add a few things to it.

In some cases, it may be important to know the current system time and date, so let's add them to our prompt display. In addition, we can add some text (called a literal) to the prompt display; this may be particularly useful if you are setting up a system for someone else.

Figure 2 shows a more complex PROMPT command. In order to emphasize the importance of spaces in the command, I have purposely left them out of the time and date commands to show how the resulting display will be affected. You can improve the readability of the information by adding spaces before and after both equal signs in the command.

Note the double $_$_ (carriage return/line feed—CRLF) sequence following the $N parameter. The first CRLF ends the line "Enter Command for Drive $N". The second CRLF gives us a blank line between the literal and the ==> prompt indicator. You can use this idea to improve the readability of your prompts.

Using the version parameter

If you have a number of system disks that were generated by using MS-DOS's FORMAT command with the /S (system) option, it is usually nice to know which version of the operating system you are using. This is especially true if you have received an update to MS-DOS, and you may not have updated all disks with the new programs.

Adding the version parameter to the prompt will always inform you which version of MS-DOS you are using. It is an easy way to display the information you would get from the VER command.

If you take a look at Figure 3, you will note that I have used a valid null character, $A, followed by a number of spaces. The purpose of the spaces is to right justify the literal "Version is: " on the screen.

The null character is useful (and in fact required) in this case—since we do not

```
PROMPT $P $Q$Q$G

A:\ ==>

A:\SYSTEM ==>

A:\SYSTEM\DOS ==>
```

Figure 1. Three possible prompt displays produced by the PROMPT command shown at the top. In the first case, the root directory is the current directory; in the next, the SYSTEM subdirectory; and finally, the DOS subdirectory within SYSTEM.

have any other value to input immediately following the PROMPT command. If we simply entered spaces, without the null character, the command would only skip the spaces until it found the first character or command parameter.

The Zenith MS-DOS documentation also describes the use of the $S parameter that allows you to have a leading space (i.e., a space displayed as the first character). This is really not an "official" MS-DOS parameter—it is simply a null character as described above.

With a few exceptions, I have covered the most common parameters that can be used with the PROMPT command. The parameters for the vertical-bar symbol ($B) and the less-than symbol ($L) are used just the same way as the $Q (equal sign) and $G (greater-than symbol) that we have already covered.

In the nearly three years that I have worked with Z-DOS, MS-DOS, and PC-DOS, I have still not been able to develop a use for the $H (backspace) parameter. There are, of course, some trivial examples for using a backspace function, but I have been able to think of nothing that provides a useful demonstration of it.

For those of you waiting to find out how to do magic things with the PROMPT command, such as silencing the H/Z100 keyclick or changing colors, I have saved that for last.

(Speaking of silencing the keyclick, the H/Z150 provides an easy way to quiet the beast—use the ALT-ESC sequence as a "toggle." That is, to silence the keyclick, just hold the ALT key while simultaneously pressing the ESC key. Similarly, use the ALT-ESC sequence to reinstate the keyclick.)

```
PROMPT Time=$T$_Date=$D$_Enter Command for Drive $N$_$_$Q$Q$G

Time=18:11:42.37
Date=Mon  8-26-85
Enter Command for Drive A

==>
```

Figure 2. The PROMPT command shown (top) will produce the five-line prompt shown here.

```
PROMPT $A        Version is: $_$_$V$_$_$P $Q$Q$G

        Version is:

IO.SYS Version 2.00
MS-DOS Version 2.13

A:\directory ==>
```

Figure 3. The PROMPT command shown at the top will provide the version numbers of the IO.SYS and MS-DOS being used and the name of the current directory.

Z100	Z150
FILES = 10	FILES = 10
BUFFERS = 15	BUFFERS = 15
	DEVICE = ANSI.SYS

Table 2. CONFIG.SYS files for the Z100 and Z150. The FILES and BUFFERS references are optional, but the Z150 requires the DEVICE specification in order to enable PROMPT to use video-control features.

A quick look at CONFIG.SYS

Before we can effectively use the ESCape sequences on the H/Z150, we must add (or create) some information for the CONFIG.SYS file. This is a text file which contains system-configuration information that is used by the system after booting.

(CONFIG.SYS is another example of an ASCII file that must be created with an appropriate editor. As I said before, EDLIN is a perfect choice since it is quick and easy to use. It is also faster than trying to use WordStar.)

H/Z150 owners must copy the ANSI.SYS file to the root directory of the system disk. ANSI.SYS essentially provides a "translation table" between American National Standards Institute ESCape sequences and sequences expected by the hardware. Use of the ANSI sequences by programs (and by MS-DOS itself, as in PROMPT) provides a measure of video I/O compatibility between all MS-DOS systems. It is very slow from a performance and user-response perspective; however, ANSI.SYS must be implemented to access the video I/O features when these features are used in the PROMPT command.

Table 2 contains the appropriate information for the CONFIG.SYS file.

Note that I have also included a version of CONFIG.SYS for the H/Z100 although it is not required in order to use the ESCape sequences. The BUFFERS and FILES parameters for both machines were added because I saw that recommendation on the ZDS Software Consultation Bulletin Board (616/982-3503).

The BUFFERS parameter sets the number of sectors the operating system uses as a disk buffer. Apparently, it should be increased to 15 for some performance improvement. (The default is 2.)

I have also increased the FILES to 10 (from its default of 8) since I have a few programs that do not seem to work properly on my H100 without the increase. The Z100 MS-DOS version 2 manual indicates that the FILES parameter is related to function calls under Xenix, Microsoft's Unix-like operating system. The manual (page 9.19) says that FILES sets "the number of open files that may be accessed by Xenix system calls"; however, that is not totally accurate from a technical perspective. It would be better to refer to "Xenix-*compatible* system calls."

MS-DOS version 2 provided many new features—among which were additional calls for operating-system functions to be used with subdirectory processing and memory allocation. New function calls—which happen to be compatible with their Xenix counterparts—include those ranging from 39H (create a directory) to 4DH (retrieve return code of a child).

The fact is that any MS-DOS program which uses these system calls *requires* that the FILES parameter be appropriately specified. If a program requires the maximum number of open files to be eight (the default) or fewer, the user does not need to update the CONFIG.SYS file. Otherwise, the CONFIG.SYS file must include a larger FILES parameter; this is usually specified in the program documentation.

Adding the information above to the configuration file should not interfere with any of your existing software; but if it does, all you have to do is rename the file (e.g., CONFIG.XXX) and reboot your system.

Using ESCape sequences

Since you may not have handy a list of the appropriate ESCape sequences for your system, I have included a list for both the H/Z100 and the H/Z150 as Table 3. A quick glance will show that the number of ESCape sequences for the H/Z150 is somewhat limited, but that is

a direct result of its IBM compatibility. The IBM Personal Computer (like the H/Z150) simply doesn't have some of the hardware features that the H/Z100 has.

Please be sure that you are using the appropriate value for all ESCape sequences. As previously mentioned, either $E or $e can be used as the ESCape character, but you must use the correct upper- or lowercase value for the other characters in a sequence.

Function	Z100	Z150
Erase CRT	$EE	$E[2J
Reverse Video		
On	$Ep	$E[7m
Off	$Eq	$E[0m (*See Note 1.*)
Color Display	$Em<fore><back>	$E[<fore>;<back>m
	foreground/background	foreground background
	0 = Black	Black = 30 40
	1 = Blue	Red = 31 41
	2 = Red	Green = 32 42
	3 = Magenta	Yellow = 33 43
	4 = Green	Blue = 34 44
	5 = Cyan	Magenta = 35 45
	6 = Yellow	Cyan = 36 46
	7 = White	White = 37 47
Keyclick Off	$Ex2	n/a
Block Cursor	$Ex4	n/a
Non-blinking Cursor	$Ex;	n/a
Keyboard Key Reassignment	n/a	a) $E[<key1>;<key1x>p <key1> is existing decimal value. <key1x> is new decimal value. b) $E[<key-code>;"string";...p (*See Note 2.*)
25th Line		n/a
Save Cursor	$Ej	
Enable 25th line	$Ex1	
Cursor Address	$EY<line><column>	
Clear end of line	$EK	
Disable 25th line	$Ey1	
Return Cursor	$Ek	

Note 1: This command turns all previous attributes off.
Note 2: The ellipsis (...) indicates that additional values may be input.

Table 3. ESCape ($E) sequences which may be used by the PROMPT command. Here and elsewhere in this article, terms enclosed in angle brackets, < and >, indicate user input; the angle brackets themselves are not typed in.

Function	Z100	Z150
Inverse Video	PROMPT EpP QQGEq (PROMPT EqP QQGEp)	PROMPT $E[7m$P QQGE[0m (PROMPT $E[0m$P QQGE[7m)
Keyclick Off	PROMPT $Ex2$N$G	n/a
Non-blinking Block Cursor	PROMPT $Ex4$Ex;NG	n/a
Color (Blue on Yellow)	PROMPT $Em16$P QQGEm40	PROMPT $E[34;43m$P QQGE[0m

Table 4. Some fancy PROMPT commands.

Developing fancy PROMPT commands

Now that we know something about the command and its syntax, it is time to do some fancy things with the existing examples. Refer to Table 4 for the following examples.

The command prompt can be displayed in reverse video by using the appropriate ESCape sequence at the beginning and end of just about any PROMPT command. Remember that you must turn the reverse video on at the beginning and off at the end of the command. If you forget, you will not hurt your system, but you will get a strange display.

For an interesting effect, try reversing the locations of $Ep and $Eq for the H/Z100. A similar command is shown for the H/Z150 in parentheses below the basic example.

The main result of both commands is to display everything *except* the command prompt in reverse video.

Some people find that the keyclick is particularly annoying on the H/Z100. Although you can write a short assembly-language program to silence the noise, it is much easier to use the PROMPT command. In the example shown, the keyclick is turned off, and the normal command prompt is displayed.

As a personal matter, I prefer a non-blinking block cursor for all of my work. I find that the underline cursor is too small to be seen well for a lot of writing, and the blinking is annoying to me.

It is a simple matter to set up a quick command line to change that cursor display. I find this particularly useful since some software developers apparently feel that they must use the ESC z power-up reset sequence (which restores the blinking, underline cursor) when entering the program or returning to the MS-DOS command prompt.

If you have a color monitor, you can also have a nice color contrast for your prompt display. My personal preference is a yellow background with blue characters. The last example in Table 4 shows you how to set up a color change for the prompt display. Refer to Table 3 for other valid colors for your system.

For H/Z100 owners only—the 25th line

Many times, we may take for granted that the last line on the screen will be inaccessible to the ordinary user—only applications programs can or need to access the 25th line. However, the H/Z100 has some control sequences which also allow you to use the 25th line for display purposes.

You might want to place the time and date on the 25th line, with the drive/directory display kept as part of the normal prompt. Figure 4 shows the appropriate form of the PROMPT command; it is really a variation of the example shown in Figure 2. But there are a few twists.

Before you can do anything with the 25th line, you must save the current cursor position with an ESCape j. In order to access the 25th line, you must enable it with an ESCape x1. The "$EY8 " (note the space following the 8) positions the cursor at column 1 of the 25th line.

Since you never know what may have been on the line, you should always clear it: an ESCape K clears the line from the cursor position to the end of the line. Return to the previously saved cursor position with an ESCape k (be sure to use

```
PROMPT $Ej$Ex1$EY8 $EKDate = $D  <40 spaces>  Time = $T$Ek$P $Q$Q$G

A:\directory ==>
.
.

Date=Mon  8-26-85                                       Time=18:11:42.37
```

Figure 4. On the Z100, the PROMPT command shown (top) will provide the current directory name at the ordinary cursor location; it will also display the time and date at the bottom of the screen on the 25th line.

Row or Column	Char	Column	Char	Column	Char	Column	Char
1	<SP> *(1)*	26	9	51	R	76	k
2	!	27	:	52	S	77	l
3	"	28	;	53	T	78	m
4	#	29	<	54	U	79	n
5	$	30	=	55	V	80	o
6	%	31	>	56	W		
7	&	32	?	57	X		
8	'	33	@	58	Y		
9	(	34	A	59	Z		
10	)	35	B	60	[		
11	*	36	C	61	\		
12	+	37	D	62	]		
13	,	38	E	63	^		
14	-	39	F	64	_		
15	.	40	G	65	`		
16	/	41	H	66	a		
17	0	42	I	67	b		
18	1	43	J	68	c		
19	2	44	K	69	d		
20	3	45	L	70	e		
21	4	46	M	71	f		
22	5	47	N	72	g		
23	6	48	O	73	h		
24	7	49	P	74	i		
25	8	50	Q	75	j		

Note 1. <SP> is the ASCII space character, " "

Table 5. Ordinary ASCII characters used to specify row and column values in direct cursor addressing with the PROMPT command. (The decimal ASCII value of these characters is equal to the number given here plus 31.) For direct cursor addressing, the PROMPT parameter takes the form $E<row><column>.

lowercase), and write the remainder of the command.

Due to space considerations, I have not shown the exact number of spaces which will allow the time to be displayed on the right hand side of the 25th line; I have found, though, that 40 spaces seems to be about right. You can also add reverse video or color to this command if desired.

I used the H/Z100's capability for direct cursor addressing in this example because it shows that you can use any valid ESCape sequence to perform a desired operation. Direct cursor addressing can be used to place the cursor anywhere on the screen, and it is a very simple technique which can also be used in any programming language, such as BASIC or assembler.

The thing to note about direct character addressing (the ESC Y sequence) is that it uses ordinary ASCII characters for line and column values. (I prefer to use decimal numbers for discussing lines and columns, so all of the following numbers will be in decimal.)

There is an easy way to determine the line and column values for the ESC Y sequence. Simply add the column or row's decimal value to 31. The resulting decimal value is the number of the appropriate ASCII character for use in the command.

Direct cursor addressing is not the main subject of this article; so, for those of you interested in this feature, I have included a translation table as Table 5. You can use this feature to create some rather spectacular prompts on the H/Z100, especially if you take advantage of the graphics capability.

Checking Table 5, you will see that 1 converts to a space, 2 to an exclamation mark, 3 to a quotation mark, and so on. If we use the PROMPT command format, the upper left hand corner (row 1, column 1) of your screen is indicated by "$EY ". (Note the two blanks.) Line 2, column 1 is "$EY! "; line 3, column 1 is "$EY" "; and so on. Since the last normal position on the screen is line 24, column 80, you can use this information to show that the correct command for that position is "$EY7o". Following this pattern, it is easy to see that line 25, column 1 must be addressed as "$EY8 ".

One interesting situation that can occur as you are writing various PROMPT commands is an "Out of environment space" error message. That error message is normally associated with the SET command, which allows you to set certain values that may be used by programs or batch files of MS-DOS commands. I received this error message, though, as I was working on a PROMPT command for the 25th line.

Well, one of those items which may have a value SET for it is PATH, which specifies directories to be searched when a program is called without a device specification and can't be found in the current directory. Another item is PROMPT.

The error message occurred because I have a very long PATH command set up for my hard disk, and the basic limit for the "environment space" is about 200 bytes.

If you get that same error message, just enter the SET command; you will then be given all of the information currently stored in the environment space, including the PROMPT command if you have entered one of these examples.

In my situation, I entered PATH \ which reset the system to "No path", and was able to finish with the command.

(By the way, all of this command information is given in my book, *The Flip-Fast Guide to: Zenith/Heath MS-DOS (ZDOS)*. It's available through *Sextant*.)

For H/Z150 owners only

One of the more unusual features of the H/Z150 (and IBM PC) is that the keyboard keys can be reassigned by means of the PROMPT command.

In the simplest case, you can change the location of the keys on your keyboard. For example, you can assign the "A" key

to generate "Q" and the "Q" key to generate "A". You must reassign both keys, or you will have lost one of the letters. Since each key generates also a lowercase letter, you will probably want to keep them consistent; so, you must remember to change the "a" to "q" and the "q" to "a".

The following PROMPT command performs those changes in exactly the order listed:

PROMPT $E[65;81p
(To change the A key to Q.)
PROMPT $E[81;65p
(To change the Q key to A.)
PROMPT $E[97;113p
(To change the a key to q.)
PROMPT $E[113;97p
(To change the q key to a.)
PROMPT NG
(To reset the standard prompt.)

The basic form for key reassignment under the PROMPT command is included in Table 3.

Each key reassignment consists of two key codes in decimal form. The first value (key1 in Table 3) of each pair is the normal code—65 is the decimal value for the ASCII A. The second value (key1x) is the reassigned code for that key—81 for the ASCII Q. The "p" indicates the end of the mapping sequence. The lowercase values for both letters are reassigned in a similar way.

The last command above is required because the first PROMPT command in the series will cause the existing command prompt and cursor to "disappear." Note that the command PROMPT NG is exactly equivalent to the PROMPT command with no optional parameters.

Reassigning keyboard keys, sometimes called mapping or remapping, can be done easily using PROMPT. But that is not the most effective use of the command unless you wish to make just small changes in the keyboard layout.

This particular ESCape sequence (ESC [) also allows you to set up special strings for your keyboard. You may find this particularly useful if you wish to press a single key (i.e., a function key) to input a command that you use frequently. Perhaps for an example, you might want to use the F10 key to generate the DIR command.

In order to use this function, you must know the keyboard scan codes (extended ASCII codes) for your system. Since these can be difficult to find in the documentation, I have shown them in Table 6.

Note that Table 6 also includes the keyboard scan codes for the shifted function keys, CTRL plus the function key, and ALT plus the function key. That effectively gives you a choice of 40 programmable function keys in case you are so inclined. This capability for function key programming is a hardware feature of the IBM PCs and compatibles. Unfortunately, most software vendors—except for SSI International, developers of the WordPerfect word processor—have not

Key	Scan Codes (Decimal): Key Only	Key Shifted	CTRL + Key	ALT + Key
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113

Table 6. Scan codes (decimal) for use in Z150 keyboard reassignment.

taken full advantage of this capability.

In Table 3, I simply used the term "key-code." You should note that the key-code includes a special definition: when the first digit in the sequence is null (i.e., zero), the first and second digits in the sequence comprise the extended ASCII code. Therefore, the key-code for the F1 key is 0;59 and the keycode for the F10 key is 0;68.

Now that we know all the necessary information, the coding of the command is easy, as follows:

PROMPT $E[0;68;"DIR";13p

In this example, the F10 key (0;68) is redefined as the letters DIR. The 13 is the decimal value of a RETURN, which you would press in order to enter a command; as above, the "p" indicates the end of the mapping sequence. You can redefine other keys in a similar manner to include such things as DIR B: or CHKDSK.

Several cautions on the use of this feature are in order.

If you decide to use this function, be sure to reset the keys before you use any software. It is unlikely that the software will reset the keys, and you may get some strange results. I recommend that you develop a second batch file (e.g., RESET.BAT) to reset all of the function keys to their original values before you use your spreadsheet or word processor.

Page 6.73 of the MS-DOS Version 2 Programmer's Utility Pack also provides a caution that you should not exceed 250 bytes of definitions: it may cause the H/Z150 to "freeze"—which requires a cold boot using the standard CTRL-ALT-DEL sequence. If your system does not respond, then you will need to resort to the power-off/on approach.

The reassignment example above uses only four bytes (DIR plus the carriage return), so you really should not have any problem so long as you don't redefine every key on the keyboard.

A note for new users

I like to encourage new users to try things on their systems, but I frequently find that they are reluctant to do so because they are afraid that they will damage their systems.

It is virtually impossible to damage your system hardware through anything that you do with any system or application program.

The only exception that I am aware of is that one form of copy-protected software can evidently damage a color monitor if you attempt to bypass the protection feature. Apparently, this is possible because the video card (not one of Heath/Zenith's) has a design flaw which permits this to happen.

There are a number of user problems associated with copy-protected software—which is one reason that I do not recommend using it. Apparently, the military agrees with that position—I just read that Lotus has finally agreed to provide them with an unprotected version of their popular spreadsheet.

If you are doing anything in the "experimental" mode, particularly programming, I recommend that you make backup copies of your disks with the DISKCOPY command.

That is always good practice, but it is especially important while you are learning the system. About the worst that can happen is that you will inadvertently erase all of the files on a disk. So long as you have a backup, you can proceed with your testing with no worries.

As indicated above, another problem that can happen is a system "freeze"—which is indicated by the fact that your system will not accept any commands at all. In some cases, this will simply require a system reboot. In others, notably on the IBM PC and compatibles, it may require you to turn the power off and on to reset the hardware.

Chances are that this will not happen to you, but it is always good to know what can happen and how to fix it.

A final prompt

In my testing of the PROMPT command, I have found that, when the $P (current drive/directory) parameter is used, some MS-DOS hard-disk commands (notably SHIP and PART) result in error messages. After they have performed their functions and attempt to return to the command prompt, they indicate that the drive is not ready when the current drive is one of the Winchester partitions.

In the case of the SHIP command, that occurs because the system is attempting to read the drive (and directory) *after* the hard disk read/write head has been positioned over the center of the disk for safer shipment.

If you encounter that problem, all you have to do is reset the display prompt with the command PROMPT before you enter any of those commands. That is also one reason why Zenith recommends that you use the operating system supplied on the Winchester Utilities Disk provided with all hard disk systems and upgrades.

(You can still use the PROMPT command with the Winchester utilities so long as you have version 2 of the utilities. The problem is that, when the $P parameter is used, MS-DOS attempts to re-read the disk directory in the default drive. If the default drive happens to be the hard disk when PART, PREP, SHIP, or DETECT is run, you will get the "drive not ready" error. If the default drive is the floppy, that will not cause a problem.)

When you finally develop the PROMPT command that suits your needs, I suggest that you include it in your AUTOEXEC.BAT file so that it will automatically be set up when you start your system. That file also must be in ASCII format as discussed earlier. I suggest that you use the following format:

```
DATE
TIME
PROMPT $N $Q$Q$G
```

All commands presented in this article were developed and tested on both the H/Z100 (MS-DOS 2.21) and H/Z150 (MS-DOS 2.13) computers. Commands shown for the H/Z150 should work as discussed for the IBM PC and other compatibles (assuming that the manufacturer has not changed the command interpreter, COMMAND.COM).

If you have any questions about the implementation of the PROMPT command based on the information in this article, please write to me at the address shown. I answer all mail within one week after receipt, but I do ask that you enclose a stamped, self-addressed envelope if you wish to have a reply.

William M. Adney
P.O. Box 531655
Grand Prairie, TX 75053

△

Performance.

Introducing EasyPC™ from UCI. The most advanced IBM PC emulator you can buy for your Z-100. First to provide the compatibility, speed and audio support to make your Z-100 run like a PC. Since the EasyPC provides complete PC circuitry, you actually run PC programs on PC hardware. Quite simply, it's like installing a PC right inside your Z-100. In the PC mode, your Z-100 can run virtually the entire library of PC programs, including LOTUS 1-2-3, dBase III, FLIGHT SIMULATOR, and most other copy-protected software. In the Zenith mode, your Z-100 is a Z-100, with no change in performance.

EasyPC is fast. In fact, it is noticeably faster than both the Z-150 and the IBM PC. And EasyPC can run at either 5 or 8 MHz. It supports both floppy and Winchester drives, provides crisp color video graphics, supports PC-compatible sound generation, and even comes with its own self mounting speaker.

Quality.

You already know and appreciate the kind of quality and reliability that is built into your Z-100. UCI matches that commitment to quality from first step to last. Sophisticated engineering and quality components combined with precision automatic assembly, component burn-in, and both in-circuit and full functional testing, mean superior performance and long term reliability.

EasyPC's advanced circuitry utilizes two, high quality, S-100 multilayer boards and a piggyback board. Yet its ingenious design requires only *one extra slot*. So you get all the advantages of PC compatibility, with room to spare.

Availability.

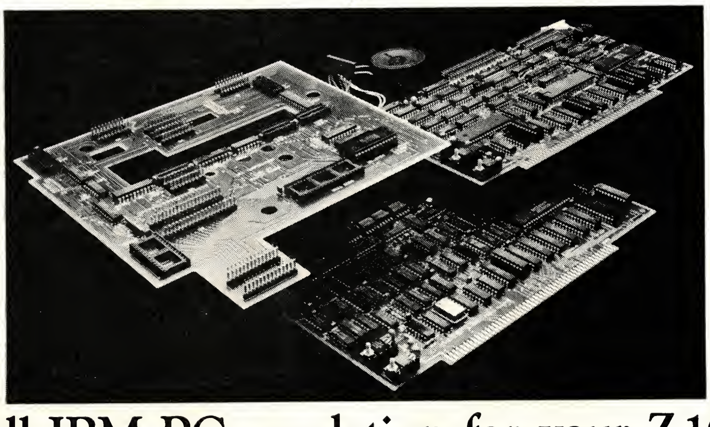
EasyPC is ready when you are. If you've been waiting too long for PC compatibility, your wait is over. EasyPC is here now. Complete documentation is included for quick and easy installation, so you'll be up and running fast. Your only problem will be which PC programs to get first.

EasyPC is available through your independent Zenith dealer. Ask him for a demonstration.

Price.

At $699, you'll pay a little more. But you'll get the best. Total compatibility. High speed performance. Unsurpassed quality and serviceability. In a word, value. You also get UCI's one year Solid Service Guarantee and full customer support. If you have any questions or problems with any UCI product, or need the name of your nearest dealer, just call 800-UCI-COMPUTER.

EasyPC™



Full IBM PC emulation for your Z-100.

UCI CORPORATION
948 Cherry Street, Kent, Ohio 44240
(216) 673-5155 / 800-UCI-COMPUTER

Circle #180 on Reader Service Card

EasyPC is a trademark of UCI, Inc. Zenith and Z-100 are trademarks of Zenith Electronics Corp. IBM and IBM PC are trademarks of International Business Machines. Lotus 1-2-3 is a trademark of Lotus Development Corp. dBase III is a trademark of Ashton-Tate. Flight Simulator is a trademark of Microsoft Corp.

C Notes

Joseph Katz

Computer Innovations' "Introducing C"

If you know somebody who wants to learn C but has been waiting until someone releases the C equivalent of Turbo Pascal, you ought to suggest a look at Computer Innovations' "Introducing C." It's available right now (but only for Z150s and such close compatibles as the IBM PC), and it looks good.

The package is one diskette and a paperback, subtitled jointly *Interactive C Language Training System.* They are just that. The book consists of a series of learning "modules"—61 in all—prefaced with a substantial section on C preliminaries. You start with a simple C program and an explanation of how to use "the editor."

By "the editor" they mean the one that's built into IC.EXE—the Introducing C program, a C interpreter. (If you're familiar with BASIC, think of Microsoft BASIC, or Z-BASIC or GW-BASIC, and its integral editor.) The interpreter is intended, and mostly seems, to be compatible with Computer Innovations' Optimizing C86 compiler. The idea is to learn the language using Introducing C, then move up to a compiler like Optimizing C86. Moving up is necessary for serious programming, in part because of course the interpreter does not produce a COM or EXE file. (Now think of Microsoft's BASIC compiler, which compiles—at least in theory—programs written and debugged with the BASIC interpreter.)

The simple things I've tried so far can be tested and debugged with Introducing C, then compiled with Optimizing C86. Introducing C's manual is not really helpful on how to use IC.EXE for that purpose, saying little more than that it can be done. There is, however, a really good section on how IC stacks up against the archetypal C language of Kernighan and Ritchie. IC doesn't support `#include`, but it does incorporate many integer functions; `#defines` must be simple and not involve tokens; no inline assembly-language is accepted; and there are other limitations. So I don't suggest Introducing C as the C equivalent of Microsoft's BASIC interpreter, nor does Computer Innovations. For developing isolated functions before compiling, though, IC is indeed handy.

No matter whether IC is used as a learning aid or a pre-compilation tool, you can either enter the source code using the editor or read it from a disk file. The disk file can have been produced with another editor, such as WordStar in the non-document mode. When everything looks right, press F2 and the program runs—or doesn't. If it doesn't, the interpreter jumps back to the code and highlights the line at which it recognized an error condition, just as in Turbo Pascal. You can keep looping this way until you get things right.

I'm not far enough along in Introducing C to be sure of much, but I have no doubt at all that one can learn C more easily with it than without. I'm also sure that I really don't like the editor as much as I like the way Turbo Pascal mimes WordStar. But then I don't like a dual-mode editor: switching in and out of command mode never appeals to me. And I can't remember this editor's commands: I'd be lost without the included reference card.

Nevertheless, I am enthusiatic about Introducing C. I even like it well enough to use for recreational programming. (All right, what I mean is I like *playing* with it as much as working with it.) It's not copy-protected, so there's no fuss or worry about using it. It's fast enough so that IC could stand for "Instant C." And it's darned good fun. It has graphics and color and sound and . . . . You get the point.

Genie and Whiz as Z100 programming tools

SideKick and its descendants now are advertised as "desktop utilities": RAM-resident programs that will keep track of such things as notes, names, and addresses.

But the way SideKick is supposed to have originated was as a tool for use in-house by Borland's programmers. That's why most desktop utilities have an ASCII chart—which puzzles business users who buy these programs for the advertised purposes. Still more puzzling to those people are special functions in the calculator—such as hexadecimal and binary arithmetic.

If you're chuckling about "throwing away money for an ASCII chart when good ones abound in books," or if you're about to crack the joke about "spending more than $50 to turn an expensive computer into a $5 calculator," don't. They mean you probably haven't worked with one of these programs—at least not with one of the good ones.

A good electronic ASCII chart shows all 256 characters, just the way they appear on the terminal, along with the codes that produce them. Approximations on paper charts simply can't substi-

tute, and certainly they're not there where and when you need them—which is on the screen while you're trying to fix a glitch in a graphics program, for example.

A good programmer's calculator, like my Texas Instruments Programmer, costs $75. But while it does well the things I can't do in my head, it has tendencies to roam. It's rarely around when I want it.

So, soon after SideKick appeared in 1984, I joined the growing ranks of Z100 owners who lusted after something like it to run on our machines. SideKick won't: it's for IBM PCs and "true" compatibles.

We turned out to have hit pay dirt without SideKick. There are several similar programs available for the Z100 now, and the two I've used are both richer than their prototype. Interestingly, both programs were written in C.

Whiz from Software Wizardry is the more limited program, especially in its calculator that works only in decimal radix. But it does have the same good electronic ASCII chart as the best of the breed.

And Whiz is unique in having two functions that can rescue the situation when a program behaves badly. First, if the video display has been wiped out, hit <BREAK-9> and the terminal will be reset to the startup condition. Second, if the cursor hangs up on line 25 and you suddenly have an 80-column by one-row display, hit <BREAK-8> and the cursor will be homed. Before Whiz, an angry reboot was the usual way to confront fouled video or having all output scroll unreadably on the Z100's 25th line. That "solution" was an admission of failure because it took time and removed all possibility of examining RAM to see what went wrong. With Whiz in RAM, a couple of keystrokes restores the screen after either problem without disturbing memory contents. SideKick doesn't have those features; neither do most of its descendants.

Advanced Software Technologies' Genie doesn't, but in all other ways it's probably the most versatile desktop utility I've seen. Not only does it have a fine programmer's calculator, but it also has an integer-only calculator that can be loaded instead when one wants simpler calculations that cost less memory.

Since all Genie functions are modular, you can make custom configurations to meet different needs. Put on your programmer's cap and you can have the full-functioned calculator, ASCII chart, notepad, and "cut and paste" features. Switch to your fedora for proper business attire and you can use the integer calculator, "Rolodex" utility, and appointment calendar along with additional modules you'd like available.

One unique feature of Genie is the capability to unload everything from RAM when you'd like: all other desktop utilities I've seen require rebooting the computer when a big program needs more RAM.

Another apparent Genie exclusive is its keymapper module, which allows assigning macros to a reasonable number of keys. If you're one of us with an unenviable record for mistyping commonly used sequences, you've yearned for a keyboard redefinition program on the Z100—something like Software Research Technologies' SmartKey—to emit them at a single keypress. And if you're as unlucky as I am, you might even have found a copy of SmartKey for the Z100 only to discover it works so badly as to be practically useless. Well, Genie's keymapper works and is useful when programming. Keep it in RAM when you program and you need never type a commonplace term like `printf` again.

To make things sweeter still, Whiz and Genie are products of reputable companies that give excellent support. You likely know Software Wizardry. Advanced Software Technologies is a relative newcomer to the Heath/Zenith community, but it has made a promising start toward a good reputation. I bought my Z100 Genie at HUGCON; a few weeks later, the mailman brought an unexpected replacement disk with an entirely revised Genie. Not only were bugs fixed, but there were new modules and improved functions. And no one asked for a penny in return!

When I begin a programming session on the Z100, I load Genie. If I get to a point where something might do naughty things to the video RAM, I unload it and load Whiz instead. (Batch files make this kind of manipulation easy.) Genie is available for the Z150, too, but I don't have it yet. There I'm stuck with SideKick.

(For more on desktop utilities for the Z100, see "Software Roundup: Five Desktop Utilities for the Z100," by David R. Felstul, in the January-February 1986 issue of *Sextant*.)

FASTIO problems?

The MS-DOS version of WordStar and other programs that handle the Z100's video as it does are unspeakably slow, so I do just about everything anybody suggests to speed things up.

When I saw the *REMark* announcement of Pat Swayne's FASTIO, a RAM-resident program for adding still more zip, I placed an early order for HUG disk 885-3025-37 containing it. After what I thought was a reasonable testing period—a whole day—I added to my Z100's AUTOEXEC.BAT the command to load it.

I soon discovered that FASTIO didn't like some programs I produced with some compilers, notably Computer Innovations' Optimizing C86. Or maybe *they* didn't like *it*. Those programs would start to run, then hang with a blank screen. A CTRL-C broke the hangup every time, but for a while I was beginning to think some squirrels had moved into either my computer or my head. Even the simplest things were misbehaving. Of course I concluded I had become terminally sloppy, which led me to start focusing on possible goofs in the code I was writing. Nothing: it looked great; it just didn't work. When I tried recompiling some old code and had the same bad results, the time came to look elsewhere.

And the finger came to rest on FASTIO. Then the solution was a snap. Because Pat knows that FASTIO does conflict with some other programs, the disk includes SETFAST—a program that can defang FASTIO. The command is SETFAST D. If you use FASTIO and are having problems similar to mine with your compiled programs, see if things change after you do a SETFAST D.

ID.C: This program knows its place

If you own a Z150 with a late ROM release and would like to see what the '150 thinks it is, run ID. (The source is in Listing 1.) You'll learn you have an IBM PC-XT.

Not really, of course. What you're seeing is one reason the Z150 runs so much software written just for the IBM Personal Computer family: it prevaricates through its ROM, telling programs that inquire that it is what it isn't. (Not so the Z100.)

First, the Z150 ROM release date is in the same eight bytes of ROM reserved by IBM for release dates in its Personal Computer ROMs. They're at locations F000:FFF5 to F000:FFFC. If you look at the ROM BIOS listing in Section 5 of the *Technical Reference* for the IBM PC (IBM Part Number 6322507), you'll see the release date there. The Z150 release date is in just the same place.

What I can't find from an admittedly quick look through the listing is the so-called "machine ID." That's a single-byte code IBM uses to identify its machines. But it's easy to find in the ROM BIOS itself, at location F000:FFFE. Peter Norton's *Programmer's Guide to the IBM PC* (pp. 60-61) has the only published discussion I've seen of this byte and its interpretation. Norton says that the value of the byte is only a rough indication of the machine itself, but is good enough for programs that need to know whether or not a machine has a particular ROM *feature*. It's a checkpoint.

For that reason it's useful in distinguishing other machines, too. As an example, Norton has "unofficial" machine IDs that can be used to discriminate between two Compaqs: the standard IBM PC model, and the XT-like Compaq-Plus. At least some early Z150s report a similarly individualistic machine ID, but the 2.0 ROM in my Z152 gives the same ID as the XT and the Portable PC.

```
/*
**  id.c -- reports the machine ID and release date in pc roms
**  (MS-DOS) only
**
**  version 1.00
**
**  from the C86 bbs, original author unknown
**  adapted by Joseph Katz for the "C Notes" column in SEXTANT
**
*/

#include <stdio.h>

main()
{
int i;
unsigned char make[8];
i = peek(0xFFFE,0xF000) & 0xFF;
switch(i) {
        case 0x2D:
                strcpy(make, "Compac");
                break;
        case 0x9A:
                strcpy(make, "Compac Plus");
                break;
        case 0xF5:
                strcpy(make, "n H/Z 150");
                break;
        case 0xFC:
                strcpy(make, "n IBM AT");
                break;
        case 0xFD:
                strcpy(make, "n IBM JR");
                break;
        case 0xFE:
                strcpy(make, "n IBM XT");
                break;
        case 0xFF:
                strcpy(make, "n IBM PC");
                break;
        default:
                strcpy(make, "n unknown");
        }
printf("The ID of this machine is %x (a%s)", i, make);
if(peek(0xFFF5,0xF000) & 0xFF) printf(",\trelease %c%c/%c%c/%c%c.\n",
        (peek(0xFFF5,0xF000) & 0xFF),
        (peek(0xFFF6,0xF000) & 0xFF),
        (peek(0xFFF8,0xF000) & 0xFF),
        (peek(0xFFF9,0xF000) & 0xFF),
        (peek(0xFFFb,0xF000) & 0xFF),
        (peek(0xFFFc,0xF000) & 0xFF));
exit(0);
}    /* end of id.c */
```

Listing 1. ID.C was written for C86 running on an H/Z150 under MS-DOS. It will report an H/Z150's machine ID and ROM release date.

If you write anything dependent on the target machine's having special features, ID.C can be rewritten into a function callable in your program. As a program itself, it is useful for checking the machine IDs and ROM release dates of computers you own or to which you have access—Heath/Zenith or not.

As a matter of fact, I volunteer to accumulate any machine IDs not in ID.C if you'll send them to me on a *postcard* or *index card*, care of Sextant. Should anything interesting turn up, I'd be happy to publish it here for the general good.

The program is based on code downloaded from the Computer Innovations Bulletin Board, which is a private BBS for members of the C86 User's Group. As presented here, I've supplemented it with IDs from Norton's book.

Manx Aztec C notes

I started using Aztec CII (the CP/M version) a few years ago on a TRS-80 Model II when I wanted a C compiler that had siblings for other systems—especially TRS-80 Models III and 4, and the Apple II—to make porting code easier. Aztec fit the bill, did the job, and made other cliches appropriate. Moreover, it was fast and produced tight, small programs. Nice.

I also was impressed when, a few weeks after I began using the package, I called Manx about an update from the version 1.6B I had bought to the 1.6D I saw advertised. No problem; no charge; the update came about a week after I sent in the distribution disk. Nicer still.

In a while, Manx began advertising a "Professional" version with all sorts of additional things that somehow did not seem worth $150 more. So when I bought Aztec C86 (the MS-DOS version) for the Z100 and Z150, I ordered what now was called the "Basic" version, which again was the lowest-priced version. And I was happy with it, too.

A few months ago, Manx burst my bubble by sending me the "Commercial" versions—the top-of-the-line when the package names seemed to change again—of both the CP/M and MS-DOS packages. Gad. Yes, indeed, for anyone above the bare novice level they *are* worth their cost. You needn't be a software developer, either, to justify the purchase. Each package has valuable extensions and features in addition to some cute whistles and bells. For one thing, both come with library source code. For another, both come with the tools needed to produce ROM code. For a third, both provide for symbolic debugging: with Digital Research's SID in the CP/M version, and with Aztec's own debugger in the MS-DOS version. I could live without the ability to make ROMable code, but I wouldn't voluntarily surrender the library source or the support for symbolic debugging. And there is more in each package.

In the MS-DOS Professional version, for example, there is an absolutely lovely editor called Z. (If you know C on Unix systems, it's supposed to be like VI.) There is a bug that makes Z on the Z150 give a "Not enough memory" error and abort. A call to Manx's support people produced a small patch that fixed the bug. Rename Z.EXE to Z and use DEBUG to patch location 6BCA (which reads "75 09 5B") to "5B 75 08." Then rename Z back to Z.EXE. Run Z and you'll see a splendid C programming editor in action.

Ecosoft C notes

Eco-C88 (MS-DOS) has a notable error-reporting feature. In addition to the cryptic error *numbers* common to most C compilers, there are pretty good error *messages* on the order of "an opening brace was expected instead of " whatever it found instead. That's a bit clearer than the way some other compilers point out errors. What's notable, though, is that many of the messages give a page number for further reference. I can't find anything in the compiler documentation—the slim Eco-C88 manual or the release notes on the package's disks—to point the way. My guess is that these are page numbers in Jack Purdum's *C Programming Guide* (Indianapolis: Que, 1983). (Purdum is in charge of Ecosoft.) Much of the time there is an explanation of those error messages on those pages of the *C Programming Guide*, but often there is not.

The Eco-C88 compiler is very nice indeed, but I think that the documentation is awful. It's just not clear. Even so, it's an excellent idea to have the compiler point

to a tutorial on how to avoid what went wrong. The manual calls out for an experienced editor to do a thorough revision.

Eco-C88, by the way, is a full-featured compiler at $49.95—down from $250! Source code for the libraries is available for $15—if you waive support on them. The proviso seems quite reasonable: should you fall into a bottomless pit while tinkering with library functions, you can't really expect Ecosoft to custom-build you a ladder for free.

Don't order Eco-C—the CP/M version—for your H/Z100. It won't work. There's nothing at all wrong with this fine compiler package. In fact, it's a favorite of mine on my Kaypro 4-84: it produces fast, tight code, and the optional SLR Systems macro assembler/linker package that costs a reasonable $49.95 more (when ordered from Ecosoft with the compiler) is well worth the money because of the things it can do that Macro 80 can't. But Eco-C is for the Z80 microprocessor *only* and *can't* be used on an 8080/8085 machine. Pity. Ecosoft is forthright if you let them know your computer when you try to order, but everyone errs one time or another. That's the reason for debuggers. So to save everyone trouble, the warning seems worthwhile here. Once again: there's nothing wrong with either the company or the product; Eco-C—the CP/M version—is designed and sold to run on the Z80 *only*.

The Software Toolworks C notes

There's a graphics toolbox for Toolworks C (MS-DOS) now in beta test. That's all I know, but not all I want to know: The Software Toolworks' quality at The Software Toolworks' prices always means a bargain is at hand.

C book notes

Terry A. Ward's *Applied Programming Techniques in C* (Glenview, Illinois: Scott, Foresman, 1985) should interest users of BDS C (CP/M) very little, others not at all. It essentially is a collection of programs in that C dialect. The explanatory text gives a kind of running commentary on the programs, but I don't find it especially useful because there's no sense of movement towards a goal. The whole thing is too fragmentary to be of much use.

All the source code is easily available in the public domain. Some—for TELEDIT, OTHELLO, and CMP—comes on the BDS C distribution disks themselves: if you want hard copy, run it off on your printer. Instead of buying this book, you'd be better off getting all of them from BDS C, the C User's Group, or a bulletin board such as CompuServe's CP/M SIG (GO PCS47, then to XA 2).

Part of the reason I say that is because you can save yourself a great deal of typing; another part is because the proofreading of this book was so badly done you can't trust the listings. For example, in PNUM.C (a line numberer), twelve lines of code have three fatal errors: no declaration of `argc` as an `int`, failure to declare `argv` as an `array`, and misdeclaration of `fd` (which should be a pointer). Typos abound throughout, so if you're a novice you'll go mad wondering what you did wrong when you can't get the program to compile. With Ward's book, it's probably the book that's wrong—not you.

You can't learn anything here except how not to do a book. Spend your money on something else.

After you say "Hello": software tools

Here's the beginning of a five-part miniseries that shows how to apply the concept of "software tools" to construct "filters," the basic kind of software tool. A wonderful book on both topics is Brian W. Kernighan's and P. J. Plauger's *Software Tools*, but its examples are in RATFOR, not the C programming language. Anyway, its audience is software engineers and intermediate programmers, not folks who have studied an introductory book or two and wonder "What do you do after you say 'Hello'?" This miniseries is for those people.

I assume you have a pretty good idea of what the introductions offer and want guidance on how to move forward, but aren't yet an accomplished C programmer. If that's indeed your stage of learning C, you probably don't want either a bunch of canned programs that improve nothing but your transcription skills, or a sequence of academic exercises that do nothing worthwhile.

This miniseries, therefore, starts you building useful software tools as stations on the way to a fairly complex program. The program is a sophisticated WordStar-to-ASCII converter—better, I think, than any other I've seen. That's why I wrote it originally. So you'll wind up with something truly useful as you learn reasons for what you're doing. Then we'll reuse our old tools in combination with new ones to build other useful programs after the miniseries is finished. Sometime around then, is my hunch, you'll begin exploring your own paths with the software tools you get here.

But you won't have to postpone gratification too long. Software tools really are single-function programs. Ours will be designed to test the function in a usable way. Once the program does its one thing well, you extract the function for reuse in combinations with other functions. And you still have use of the original program. From such small building blocks are larger, more sophisticated programs made with little effort or debugging. Obviously, this direction is one in which you should go to become an accomplished C programmer: your goal always ought to be to write generalized functions you can reuse in other programs. With luck, we should be able to do the entire series in two or three installments.

I want to get the most out of my operating system!

Send me

☐ *Zenith/Heath MS-DOS/Z-DOS*

_____ copies @ $19.95 each = $ _____

☐ *Zenith/Heath CP/M-80/85*

_____ copies @ $12.95 each = $ _____

+ $2.50 shipping charge *per order* in U.S./Canada = $ _____

or

+ $4.00 shipping charge *per book* sent overseas = $ _____

Total = $ _____

Name _____________________

Address _____________________

____

____

☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank)

☐ Charge my: ☐ VISA ☐ MasterCard

Card # _____________________

Expires _____________

Mail this card today to:
Sextant Publishing Company
716 E Street, S.E.
Washington, DC 20003

21-M/C

I want to get the most out of my operating system!

Send me

☐ *Zenith/Heath MS-DOS/Z-DOS*

_____ copies @ $19.95 each = $ _____

☐ *Zenith/Heath CP/M-80/85*

_____ copies @ $12.95 each = $ _____

+ $2.50 shipping charge *per order* in U.S./Canada = $ _____

or

+ $4.00 shipping charge *per book* sent overseas = $ _____

Total = $ _____

Name _____________________

Address _____________________

____

____

☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank)

☐ Charge my: ☐ VISA ☐ MasterCard

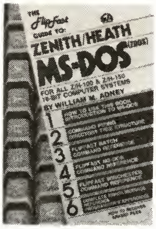
Card # _____________________

Expires _____________

Mail this card today to:
Sextant Publishing Company
716 E Street, S.E.
Washington, DC 20003

21-M/C

Send me the *FlipFast Guide* to my Heath/Zenith operating system:



☐ *Zenith/Heath MS-DOS/Z-DOS*

256 pages

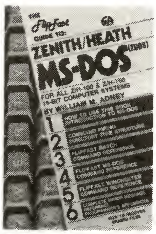
☐ *Zenith/Heath CP/M-80/85*

190 pages

Each guide features
- Easy-to-access information for Heath/Zenith users
- Specific information for each operating command or subcommand
- Easy-to-use listings and tables
- Error message listings

To order your copies, fill out the reverse side of this card today and mail it to
Sextant Publishing Co.
716 E Street, S.E.
Washington, DC 20003

Send me the *FlipFast Guide* to my Heath/Zenith operating system:



☐ *Zenith/Heath MS-DOS/Z-DOS*

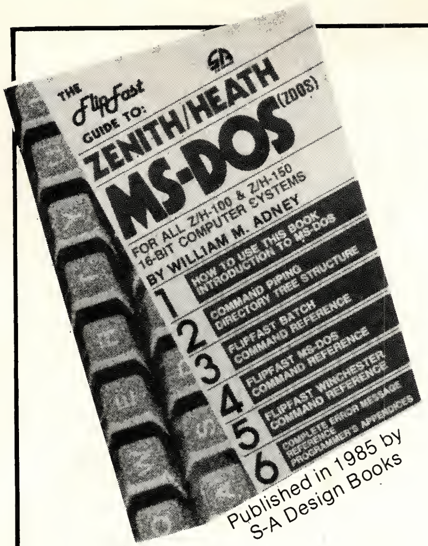
256 pages

☐ *Zenith/Heath CP/M-80/85*

190 pages

Each guide features
- Easy-to-access information for Heath/Zenith users
- Specific information for each operating command or subcommand
- Easy-to-use listings and tables
- Error message listings

To order your copies, fill out the reverse side of this card today and mail it to
Sextant Publishing Co.
716 E Street, S.E.
Washington, DC 20003



Published in 1985 by S-A Design Books

256 pages

Z-DOS and MS-DOS Users:

Get quick access to MS-DOS/Z-DOS user commands in *The FlipFast Guide to Zenith/Heath MS-DOS/Z-DOS*

The FlipFast® Guide to Zenith/Heath MS-DOS/Z-DOS is designed to be an easy-to-use reference for the MS-DOS/Z-DOS operating system commands.

This FlipFast® guide is useful both to users familiar with MS-DOS/Z-DOS commands, and to newcomers to the operating system. Each command is clearly illustrated with examples and brief explanations of operating and functional requirements. Subcommands for batch-file processing are covered, as are the special commands specific to Zenith/Heath applications within MS-DOS (version 2.0) and Z-DOS (version 1.25).

The following is provided for each MS-DOS/Z-DOS command or subcommand:

- The **Name** of each command is listed alphabetically in the FlipFast® reference sections.
- Command **Syntax** follows the name and includes both required and optional parameters.
- A short explanation of the typical **Application** of the command or subcommand is provided.
- Useful **Examples** are detailed that include an explanation of command function.
- A list of **Requirements** illustrates correct/optimum command operation.
- **Application notes** are included about specific usage, including "quirks," tips, etc., developed from extensive testing.
- Finally, **Error messages** are listed and explained for each command, where applicable. (Many error messages presented in these lists are not covered in other literature.)

Also included in this MS-DOS/Z-DOS FlipFast® guide are • a complete cross-reference to over 500 MS-DOS/Z-DOS error messages, including a page reference to the command that developed the error • a reference to EDLIN and all the EDLIN subcommands • a decimal/hex/ASCII/control-key/MS-DOS-call conversion • a glossary of terms

Order your copy today! If you're not completely satisfied with either book, just return it to us within 30 days for a full refund.

Please enclose payment with order. We accept checks (in U.S. dollars payable on a U.S. bank), money orders, Visa and MasterCard.

Fill out the coupon below, and mail it with payment to:
Sextant, Dept. 21, 716 E Street S.E., Washington, DC 20003

☐ Send me *Zenith/Heath MS-DOS/Z-DOS*: ___ copies @ \$19.95 ea. = \$_____

☐ Send me *Zenith/Heath CP/M-80/85*: ___ copies @ \$12.95 ea. = \$_____

+ shipping charge (\$2.50 *per order* in U.S./Canada; \$4.00 *per book* overseas) = \$_____

Total = \$_____

☐ Payment enclosed (checks must be in U.S. dollars payable on a U.S. bank)

☐ Charge my: ☐ Visa ☐ MasterCard

Card # ____________________ Expires ____________

Name ____________________

Address ____________________

____

____

Sextant, Dept. S21, 716 E Street S.E., Washington, DC 20003

CP/M Users:

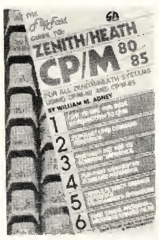
The FlipFast Guide to Zenith/Heath CP/M-80/85 is the guide you've been waiting for!

For each CP/M command or subcommand find
• required syntax • typical applications • specific examples • command requirements • application notes • error message listings

Easily refer to listings of
• CP/M 2.2 system calls 0–40 • BIOS jump table • 8080/8085 op codes • escape sequences and key functions

Plus additional technical information on
• the disk parameter block • disk parameter headers • file control blocks • CP/M memory addresses • contents of first page in memory

Features a 6-page index of error messages.



190 pages
Published in 1984 by S-A Design Books

Circle #246 on Reader Service Card

Vendors Supporting "C Notes"

BD Software
Box 9
Brighton, MA 02135
617/782-0836
(BDS C)

Computer Innovations
980 Shrewsbury Avenue
Tinton Falls, NJ 07724
201/542-5920
(Optimizing C86 [MS-DOS])

Ecosoft
6413 North College Avenue
Indianapolis, IN 46220
317/255-6476
(Eco-C88 [MS-DOS])

Lattice
Box 3072
Glen Ellyn, IL 60138
312/858-7950
(Lattice C [MS-DOS])

Manx Software Systems
Box 55
Shrewsbury, NJ 07701
201/780-4004
(Aztec CII [CP/M] and Aztec C-86 [MS-DOS])

The Software Toolworks
14478 Glorietta Drive
Sherman Oaks, CA 91423
818/986-4885
(Toolworks C [MS-DOS], C/80 [CP/M])

Table 1. In the text of his column, the author indicates where different compilers may require different code in the same C program. These vendors have facilitated this process by providing him with copies of their C compilers and with support.

Filters

As for filters, Kernighan and Plauger tell it best: "A surprising number of programs have one input, one output, and perform a useful transformation on data as it passes through. Such programs are called *filters.*"

They make life easier. A problem that always plagues microcomputer users is how to take data produced by one program (or one computer) and provide it to another program (or computer) in a usable form. Anyone familiar with C programming can solve that problem easily with a filter. In one end comes the "old" data, out the other comes the "new," and the desired changes are made in between. The changes are made according to a "table of equivalences." The filter program checks each byte against the table; if it's found, it's changed to a new value specified in the table.

Filters can be plain or fancy, but the fundamental algorithm for a filter is extremely simple:

1. open the input for reading;
2. open the output for writing;
3. get bytes in order from the input;
4. check incoming bytes against a table;
5. translate any bytes found on the table;
6. put the transformed bytes in order to the output;
7. close both files;
8. stop and exit to operating system.

If you look carefully at this algorithm, you'll see it covers three component parts: input (1, 3, 7, 8), output (2, 6, 7, 8), and transformation (4-5).

Test your current understanding of the software tools concept for a moment. It ought to suggest that the best way to write C programs is by linking together little black boxes, each of which does one discrete part of the whole job. Given enough time and experience, therefore, you might be able to draw on a library of little black boxes to build one "original" program after another.

If that's your current understanding, you're right.

You should understand, too, that the details of each stage in this miniseries may require some adjustment to suit a specific compiler. Several major publishers of C compilers have generously provided me with copies and support, so I can tell you how to make the adjustments for their compilers. These vendors are listed in Table 1. If you have another compiler, I'm afraid you're on your own. So if you don't own a C compiler yet, you ought to consider seriously buying one of those supported here. Their publishers have demonstrated a ready interest in reaching out to customers, which promises good things in case you need help. All their compilers are good. No fooling.

A "filter kit"

One implication in the concept of software tools for writing filters is we ought to build them by combining three little black boxes: one for input, one for output, and one for the transformation that is the business part of the filter. Then all we have to do is change "transformation boxes" for each new filter. And that's just about what we'll do.

I've said "just about" because C allows a single function to handle both file input and file output. It's the efficiency of C—some call it "terseness"—that also makes it able to do file input/output (I/O), including control of the flow from input to output, with just two functions in the single box.

All our filters will build on that box, which really is a fully functional file-copying program. We'll call it SKELETON; it's given in Listing 2. It opens the input and output files, pumps bytes from input to output while there are any, then closes both files and stops. SKELETON is not itself a filter because the bytes are not transformed within the program. Nor is it

worth writing for its own sake, because PIP (CP/M) and COPY (MS-DOS) do the same job better. But it is extremely valuable as a software tool, because transformation functions can easily be plugged into it to make it a filter. Think of SKELETON.C as a "filter kit."

The program, as presented in Listing 2, is little more than a combination of the standard way to handle file I/O with command-line arguments and the standard way to control the flow of bytes from input to output.

For the compiler-dependent parts of SKELETON.C, check your manual and look in your compiler's `stdio.h`. Most compilers—including Manx Aztec CII (CP/M) and C86 (MS-DOS), Computer Innovations Optimizing C86, Ecosoft's Eco-C (CP/M and MS-DOS), Lattice C, and The Software Toolworks' Toolworks C (MS-DOS)—require `stdio.h` be `#included` as the very first statement; BDS C uses `#include <bdscio.h>` there instead; The Software Toolworks C/80 (CP/M) needs `#include <stdio.h>` deleted and `#include <stdlib.c>` put instead as the last statement in the program. The Software Toolworks CP/M compiler does require the two `#defines`, but its MS-DOS compiler does not. Other compilers may need one, the other, or neither. Omit both for the Manx and Ecosoft compilers (CP/M and MS-DOS); use both for Computer Innovations Optimizing C86. Most compilers that do not already `#define EOF` (the end-of-file flag) in `stdio.h` will accept the above definition; others will need `#define EOF 0` instead. If the manual doesn't tell you, trial-and-error surely will.

Once this program is compiled and linked, the command is SKELETON INPUT OUTPUT—substituting the appropriate filenames, of course. After a few seconds, depending on the input's length, SKELETON will have finished its work and you'll find the output file in the directory.

Eventually, we'll flesh out the skeleton to make it more agreeable. Next time, however, we'll concentrate on building it step by step into the WordStar-to-ASCII filter.

```
/*
**  skeleton.c -- universal i/o skeleton for filters
**
**  note:  it's a 'skeleton' for filters, not a filter itself!
**     read the "C Notes" column to get the distinction.
**
**  version 1.00 -- barebones version to be fleshed out later
**     with all sorts of errror checking and amenities.
**
**  by Joseph Katz for the "C Notes" column in SEXTANT
*/

#include <stdio.h>              /* compiler dependent */

#define EOF -1                  /* compiler dependent */
#define NULL 0                  /* compiler dependent */

int c;                          /* input bytes */
FILE *input, *output, *fopen();

main(argc,argv)                 /* accept command args */
int argc;                       /* count args */
char *argv[];                   /* array of ptrs to args */
{

/* open input if it exists */
    input=fopen(argv[1],"r");   /* open input */
    if(input==NULL) {           /* no such file */
        printf("Can't find the input file\n");
        exit(1);                /* error exit */
    }

/* open output and check for full disk or directory */
    output=fopen(argv[2],"w");  /* open output */
    if(output==NULL) {          /* disk or directory full */
        printf("Can't make the output file\n");
        exit(1);                /* error exit */
    }

/* flow control and transformation section */
    while((c=getc(input))!=EOF) { /* flow control */

        /* transformation functions go here to make a filter */

        putc(c,output);         /* write to output */
        putchar(c);             /* display on console */
    }

/* termination section */
    fclose(input);              /* close files */
    fclose(output);
    exit(0);                    /* normal exit */

}  /* end main() */
```

Listing 2. SKELETON.C is a file-copying program; transformation functions can be plugged into it in order to turn it into a filter. It will be used in this way in future columns on filters. (See text for compiler-dependent elements of SKELETON.)

Additional Information

Software Tools
by Brian W. Kernighan and P. J. Plauger
Paperback, ISBN 0-201-03669-X, $24.50.
Addison-Wesley Publishing
Reading, MA

Applied Programming Techniques in C
by Terry A. Ward
Paperback, ISBN 0-673-18050-6, $19.95.
Scott, Foresman
Glenview, IL

The Peter Norton Programmer's Guide to the IBM PC
by Peter Norton
$19.95.
Microsoft Press
Bellevue, WA

C Programming Guide
by Jack Perdum, Ph.D.
ISBN 0-88022-157-7, $19.95
Que Corporation
7999 Knue Road
Indianapolis, IN 46250

Introducing C, Version 1.11 (MS-DOS 2.0), $95.
Computer Innovations
980 Shrewsbury Avenue
Tinton Falls, NJ 07724
800/922-0169

Genie (MS-DOS 2.0 for Z100 and Z150), $49.95.
Advanced Software Technologies
452 West 47th Street
New York, NY 10036
212/247-0150

Whiz (Z100 MS-DOS 2.0), $49.95.
Software Wizardry
1106 First Capitol Drive
St. Charles, MO 63301
800/862-8948

SmartKey II Plus (MS-DOS 2.0 for Z100 and Z150), $49.95.
Software Research Technologies
3757 Wilshire Boulevard, Suite 211
Los Angeles, CA 90010
213/384-4120

MEDIA MASTER

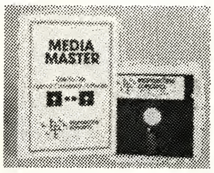
FORMATS SUPPORTED	VERSION: IBM PC	ZENITH Z-100	OSBORNE 1/EXEC	OSBORNE VIXEN	KAYPRO 2/II	KAYPRO 2x 4 10	DEC RAINBOW	SANYO 555
Actrix (DSDD)	●	●				●		●
Actrix (SSDD)	●	●	●	●	●	●	●	●
Avatar TC10 (DSDD)	●	●				●		●
Casio FP 1000 (DSDD)	●	●				●		●
Chameleon CP/M-80	●	●				●		●
Columbia MPC CP/M-80	●	●				●		●
Cromemco CDOS (SSDD)	●	●	●	●	●	●	●	●
Cromemco CDOS (SSSD)			●	●	●	●		
Cromemco w/Int'l Term (DSDD)	●	●				●		●
Cromemco w/Int'l Term (SSDD)	●	●	●	●	●	●	●	●
DEC Rainbow CP/M-86/80							●	
DEC Rainbow MS-DOS							●	
DEC Mate II CP/M (SSDD)							●	
DEC VT180 (SSDD)	●	●	●	●	●	●	●	●
Davidge (DSDD)	●	●				●		●
Digilog (DSDD)	●	●				●		●
Epson Multifont (DSDD)	●	●				●		●
Epson QX-10 (DSDD)	●	●				●		●
Facit DTC (SSDD)							●	
Fujitsu Micro 16S (DSDD)	●	●				●		●
Groupil III CP/M (DSDD)	●	●				●		●
Heath w/Magnolia CP/M	●	●	●	●	●	●	●	●
Heath/Zenith Z-DOS (1.xx DSDD)	●	●		●		●		●
Heath/Zenith Z-DOS (1.xx SSDD)	●	●	●	●	●	●	●	●
Heath/Zenith Z-DOS (2.xx DSDD)	●	●		●		●		●
Heath/Zenith Z-DOS (2.xx SSDD)	●	●	●	●	●	●	●	●
Heath/Zenith Z-100 CP/M (DSDD)	●	●						●
Heath/Zenith Z-100 CP/M (SSDD)	●	●	●	●	●	●	●	●
Heath/Zenith Z90 40 track (1k block)	●	●	●	●	●	●	●	●
Heath/Zenith Z90 40 track (2k block)	●	●	●		●	●	●	●
IBM PC CP/M-86 (DSDD)	●	●				●		●
IBM PC CP/M-86 (SSDD)	●	●	●	●	●	●	●	●
IBM PC-DOS (1.xx DSDD)	●	●		●		●		●
IBM PC-DOS (1.xx SSDD)	●	●	●	●	●	●	●	●
IBM PC-DOS (2.xx DSDD)	●	●		●		●		●
IBM PC-DOS (2.xx SSDD)	●	●	●	●	●	●	●	●
IDEA Bitelex (SSDD)	●	●	●		●	●	●	●
ISM CP/M (DSDD)	●	●				●		●
Ithaca (SSDD)							●	
Insight Dev IQ-120 (SSDD)	●	●	●		●	●	●	●
Kaypro 10 (DSDD)	●	●				●		●
Kaypro 2 (SSDD)	●	●	●	●	●	●	●	●
Kaypro 4 (DSDD) *	●	●				●		●
LNW-80 (SSDD)	●	●	●	●	●	●	●	●
Loho MAX-80 (SSDD)	●	●	●	●	●	●	●	●
Lobo MAX-80 512 (SSDD)	●	●	●		●	●	●	●
Micral 9050 CP/M-80 (DSDD)	●	●				●		●
Morrow MD11 (DSDD)	●	●						●
Morrow MD2 (SSDD)	●	●	●	●	●	●	●	●
Morrow MD3 (DSDD)	●	●						●
MacSym 350 (SSDD)							●	
Monroe OC8820 (SSDD)							●	
NCR Decision Mate 5 (DSDD)	●	●				●		●
NEC PC-8001A (DSDD)	●	●				●		●
NEC PC-8001A (SSDD)	●	●	●	●	●	●	●	●
Olympia ETX II (SSDD)	●	●	●		●	●	●	●
Olympia EX100 (DSDD)	●	●				●		●
Osborne 1 (SSDD)	●	●	●	●	●	●	●	●
Osborne 1 (SSSD)			●	●	●	●		
Osborne 4 Vixen (DSDD)	●	●		●		●		●
Osborne Executive (SSDD)	●	●	●	●	●	●	●	●
Osborne Osmosis (SSDD)	●	●	●		●	●	●	●
Otrona (DSDD)	●	●				●		●
Phillips 3000 (SSDD)							●	
PMC Micromate (DSDD)	●	●				●		●
Reynolds & Reynolds (SSDD)	●	●	●		●	●	●	●
Sanyo CP/M (DSDD)	●	●				●		●
Superbrain (DSDD)	●	●				●		●
Superbrain Jr (SSDD)	●	●	●		●	●	●	●
Systel II (SSDD)	●	●	●	●	●	●	●	●
Systel III (DSDD)	●	●				●		●
S-100 40 TRK (SSDD)			●		●	●	●	●
TI Professional CP/M-86 (DSDD)	●	●				●		●
TI Professional CP/M-86 (SSDD)	●	●	●	●	●	●	●	●
TRS-80 Model III FEC CP/M (SSDD)	●	●	●		●	●	●	●
TRS-80 Model III FEC T805 (SSDD)	●	●	●		●	●	●	●
TRS-80 Model III Hurr. Labs (SSDD)	●	●	●		●	●	●	●
TRS-80 Model III Mem. Merch. (SSDD)	●	●	●	●	●	●	●	●
TRS-80 Model I Omikron CP/M (SSSD)			●	●	●	●		
TRS-80 Model IV CP/M Plus (SSDD)	●	●	●	●	●	●	●	●
TRS-80 Model IV Montezuma Micro (SSDD)	●	●	●		●	●	●	●
Teletek 40 track (SSDD)	●	●	●		●	●	●	●
Teletek 80 track (SSDD)							●	
Toshiba T100 (DSDD)	●	●				●		●
Televideo TurboDOS (DSDD)	●	●				●		●
Televideo 802 CP/M						●		●
Wang Maws CP/M (DSDD)	●	●				●		●
Xerox 820 II (SSDD)	●	●	●	●	●	●	●	●
Xerox 820 II (SSSD)			●	●	●	●		
Zorba 40 track (DSDD) *	●	●				●		●

** Requires access to "foreign" computer*

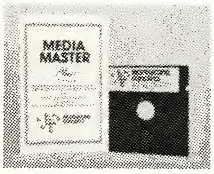
"MAKE YOUR Z-150 CP/M COMPATIBLE"

Intersecting Concepts Announces 3 Solutions To Solve Your Computer Incompatibility

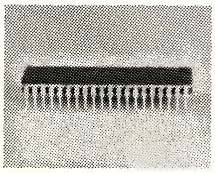
But will it work on my computer? Yes! Finally, there are three easy ways to exchange information, transfer files, and run CP/M software on MS-DOS machines.



1. **MEDIA MASTER** is our direct disk-to-disk format conversion program. Already an accepted industry standard, this $39.95 program uses simple screen prompts that lets you read, write and format up to 70 different 5¼″ diskettes from CP/M, MS-DOS and PC-DOS operating systems. So if you work on a IBM PCompatible at the office, but use a CP/M computer at home, now you can easily transfer files that would otherwise be "foreign" to your computer's operating system.



2. **MEDIA MASTER PLUS** goes one step further by converting 8-bit CP/M software to run on 16-bit MS-DOS and PC-DOS machines. This newly released $59.95 product combines our IBM-PC version of Media Master with ZP/EM, a powerful new emulation program. The results are amazing: CP/M programs using 8080 instructions *and* data can be transfered from popular computers like Osborne, Kaypro and Zenith to run on MS- DOS and PC-DOS machines!



3. **ACCELER-8/16** is also new and dramatically improves the performance of Media Master Plus by tailoring the CP/M emulation around a NEC V20 microchip. This chip simply replaces the 8088 processor in your MS-DOS computer. Once installed, it'll run your CP/M and MS-DOS software much faster. (Speed improvements are roughly 15% faster in MS-DOS and 350% faster in CP/M!) With ACCELER-8/16 your MS-DOS computer is now CP/M compatible for only $99.95!

The Choice Is Yours

So if you just need to exchange files between different computers using different operating systems, there's Media Master at $39.95.

If you're upgrading to an MS-DOS or PC-DOS machine or already have one but don't want to spend money buying "new" software programs you already own, there's Media Master Plus for $59.95.

And if the execution speed of your CP/M programs is important, there's ACCELER-8/16 for $99.95, which provides a simple hardware and software solution to making your MS-DOS computer CP/M compatible. All three solutions save you money by eliminating expensive modems and communications software.

To Order

To order Media Master, Media Master Plus, or ACCELER-8/16, call 800-824-7888, and ask for operator 251. For additional product and upgrade information contact:

Intersecting Concepts, Inc.,
4573 Heatherglen Ct., Moorpark, CA 93021
or call 805-529-5073.

INTERSECTING CONCEPTS

Circle #183 on Reader Service Card

See Seven Sorts

There are a number of ways you can go about sorting a list. Presented here are descriptions of seven internal sorts, and a program that lets you actually see them work.

Robert Crawford

The problem of sorting a list of data is certainly a ubiquitous one in computer science. It occurs at all levels—from the most mundane of data processing applications to the maintenance of tables in compiler design.

Many techniques have been advanced as correct solutions to the problem. Some of these techniques are straightforward in their operation, and others are quite subtle indeed. They also differ in the difficulty of coding, the range of application, the memory space which they require in the computer, and the speed at which they run.

Sorting methods can generally be divided into two broad categories: internal sorts, where the sorting can be accomplished entirely in the main memory of the machine; and external sorts, where the items to be sorted reside on some auxiliary storage device and are too numerous to be brought in to main memory all at once.

While those categories are not mutually exclusive, the approaches taken are necessarily quite different and the two problems are best treated separately.

In this article, I will concentrate entirely on internal sorts.

Moreover, I'll concentrate on comparison sorts: sorts based on comparing pairs of elements. (Those elements could be either letters or numbers; I'll use numbers here.) I'll present brief descriptions of the workings of seven of the most common of these sorts, tables giving comparisons of their relative efficiency (both theoretical and empirical), and a program which allows one to actually watch the sorts at work. (See Listing 1.)

This restricted view gives us a workable number of sorts, but it is only a start at looking at sorts in general. Comparing pairs of elements is only one general strategy of sorting—even though a popular one.

Long-time readers of *Sextant* may remember a quite different sort of sort: the radix sort.

(In *Sextant* #3, Fall 1982, Rick Lutowski discussed the theory behind a sorting product of his, ADSORT, and engendered a bit of controversy regarding both how revolutionary the idea was and how desirable. See "The World's Fastest Sort?" and the letters columns in the subsequent two issues.)

A radix sort uses much the same strategy that you might use to sort address cards by ZIP code. You could put all those ending with zero in one pile, those ending with 1 in another, and so on. After putting the piles in order, do the same for the next digit.

There are other strategies possible. But looking at comparison sorts is a good way to start looking at sorts in general. Moreover, they lend themselves to the graphic treatment provided by VSORTS.BAS, the program in Listing 1. (VSORTS.BAS was written for use with Microsoft BASIC running under the CP/M operating system.)

Running the program, you'll see the original, unsorted list displayed as small blocks scattered around the screen. As a given sort progresses, the individual blocks will form into an ordered row running diagonally across the screen. (The program is discussed in more detail below.)

I hope that viewing the sorts in action will further your understanding of their operation. The seven sorts (in alphabetical order) are the bubble sort, heap sort, insertion sort, merge sort, quick sort, selection sort, and Shell sort.

In the following explanations, I will refer to the list to be sorted as X(1), ..., X(N), and I will assume that the list is to be sorted into ascending order.

In the outlines which follow, line numbers mentioned refer to Listing 1, VSORTS.BAS. Table 1 gives the theoretical run times of the seven sorts. Table 2 gives some actual relative run times for the sorts on randomly ordered lists.

The workings of the sorts

The bubble sort

Oh well, I suppose I must.

The bubble sort is the first sort most of us are exposed to. (Considering the fact that it is the worst of the common sorts, it is seen much too often in application programs—in dBASE II, among others.)

A pass of the bubble sort consists of going through that portion of the list which may still be in disarray, comparing adjacent pairs of elements along the way, and switching those pairs which are indeed out of order. After a pass, at least one additional element will be in its proper place, so that at most N–1 passes are necessary—and that is exactly what happens in the worst case!

Despite the fact that bubbling is a technique best left to young children with glasses of Coke, there are bubble sorts and there are bubble sorts. The version I present attempts to be relatively intelligent as such things go. (See the

Performance	Bubble	Heap	Insertion	Merge	Quick	Selection	Shell*
Best	n	n log(n)	n	n log(n)	n log(n)	n^2	$n^{3/2}$
Worst	n^2	n log(n)	n^2	n log(n)	n^2	n^2	$n^{3/2}$
Average	n^2	n log(n)	n^2	n log(n)	n log(n)	n^2	$n^{3/2}$

*It should be noted that the theoretical analysis of the Shell sort is very difficult and the given functions represent best guesses for the version presented.

Table 1. Theoretical run times of the seven sorts discussed in the article. (See Table 2 for relative times for an actual run.)

Dr. Robert Crawford is head of the Computer Science Department at Western Kentucky University.

subroutine beginning at line 15000.)

For one thing, this version keeps a record of where it made the last switch on a given pass; it uses that information to bound the portion of the list which it must examine on the next pass. It also has the wit to realize that when no switches were necessary during a pass, it is time to quit.

Heapsort

We move now from the ridiculous to the truly esoteric. The heapsort is going to take a disorderly set of data and turn it into a *heap*.

First, what is a heap?

A heap (as in Figure 1) is a binary tree in which the following three conditions hold:

1) There is an integer k such that every "leaf" (end point) of the tree is at either distance k or distance k–1 from the root.

2) All the leaves at the lowest level are as far left as possible.

3) The value stored at a given node is bigger than or equal to the values stored at its descendants.

I told you it was esoteric.

A look at Figure 1 may clarify the definition. It shows a heap of ten elements where the appropriate value of k is 3. Note that the largest node is the root of the tree. Property 3) implies that this must be the case in any heap.

Note also that merely having a heap does not mean that our numbers will be ordered in a uniformly ascending or descending order. As you can see in Figure 1, the numbers are only "semi-ordered" by virtue of being organized into a valid heap.

Elements ordered into a heap may conveniently be stored in an array without the use of any pointers by a simple addressing scheme: the nodes of the tree are listed in order as they appear in the picture, level by level.

Specifically, we store the root of the heap in array location 1, the leftson of the node which is in array location i in array location 2i, and the rightson of the node which is in array location i in array location 2i + 1.

Thus the heap of Figure 1 corresponds to the array given in Figure 2.

The heapsort algorithm can be seen in the subroutine beginning at line 20000. It will produce an array in which the numbers are in ascending order from left to right. It proceeds as follows.

With the unsorted data in an array, the addressing convention just described allows us to treat the array as a binary tree of the shape specified by conditions 1) and 2) of the definition of a heap.

Since the array values are unsorted, this tree is not necessarily a heap. So, we must "heapify" it.

This is done by beginning with the lowest level (exclusive of the leaves) and swapping each node (v) with one of its sons, if necessary, to make the subtree rooted at v into a heap.

Comparing pairs of numbers is only one general strategy of sorting.

At each level, we proceed from left to right. If a swap causes some lower heap to lose the heap property, we reheapify it. Having obtained a heap, the largest element is at the root, so we exchange the root with the rightmost leaf at the lowest level; for the duration of the algorithm, we pretend that the new rightmost leaf at the lowest level does not exist. If the remaining tree has at least two leaves, we return to the heapifying process. Whew.

We come away with a sort of "reverse heap"—as seen in Figure 3. By the addressing scheme described above, this produces an ordered array of numbers.

The insertion sort

The insertion sort is given in the subroutine beginning at line 30000. It works by comparing a list element with the elements in the previously sorted portion of the list. The element is inserted when the correct position is found. Of course, at the beginning of the sort, only the one-element list $X(1)$ is sorted.

In general, if the first j–1 elements have already been arranged in sorted order, the jth element is removed from the list and compared successively to the previous elements $X(j-1)$, $X(j-2)$, …, $X(1)$. If the immediately previous element is larger than $X(j)$, it needs to come after $X(j)$ in the sorted list; therefore, to make room for $X(j)$, it is moved down one position—filling up the space previously occupied by $X(j)$. And so we continue, moving down the sorted list, always moving numbers larger than $X(j)$ into the most recently vacated position.

If a previous element is found which is less than or equal to $X(j)$, or if the entire list is exhausted, $X(j)$ is placed in the most recently vacated position on the list.

The merge sort

This is the only one of our seven sorts which uses additional array storage beyond that for the given list. As you can see by looking at the subroutine (beginning at line 35000), the coding is rather complicated, but the idea is simple.

At each stage of the sort, the list has been divided into a sequence of sublists each of which is itself sorted. (At the beginning, the list can be thought of as a sequence of N ordered sublists, each of which is one element long.)

Adjacent pairs of these sublists are then merged to form new sorted sublists. In merging, the first numbers of two lists are compared. The lower of the two is then placed at the beginning of a new list; the list from which this element was taken now has a new first number. Then the first numbers of the two lists are again compared. And so on.

(It is here that the auxiliary array is used to hold the newly formed sublists—they are then copied back into the original array.)

After each pass, there will be about half as many sublists as before, each of which will be roughly twice as long as before. Eventually, this process leads to a single sorted sublist which comprises the entire array.

Quicksort

This sort uses the "divide-and-conquer" technique; it is most easily programmed in a language which supports recursion. (Nonetheless, our nonrecursive version—beginning at line 45000—will be more efficient.)

The idea is that we choose an element of the array to be sorted (called the pivot); then we walk through the array making switches in such a way that all the elements which occur prior to the position occupied by the pivot are smaller in value than the pivot, and all the elements which occur in positions following the pivot are larger than the pivot. When these switches have been made, the pivot element is in the position it will occupy

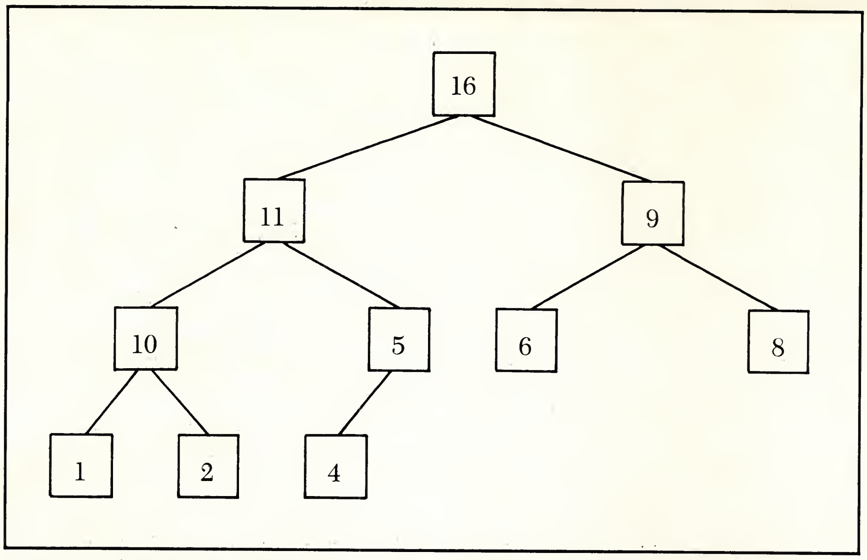


Figure 1. A binary tree in the form of a "heap." (See Figure 2 for an array holding the values of this heap.) This figure shows a valid heap, but it is not the form which would be produced after a heap sort. For that, see Figure 3.

16	11	9	10	5	6	8	1	2	4

Figure 2. An array containing the values of the heap shown in Figure 1, stored according to the convention discussed in the text.

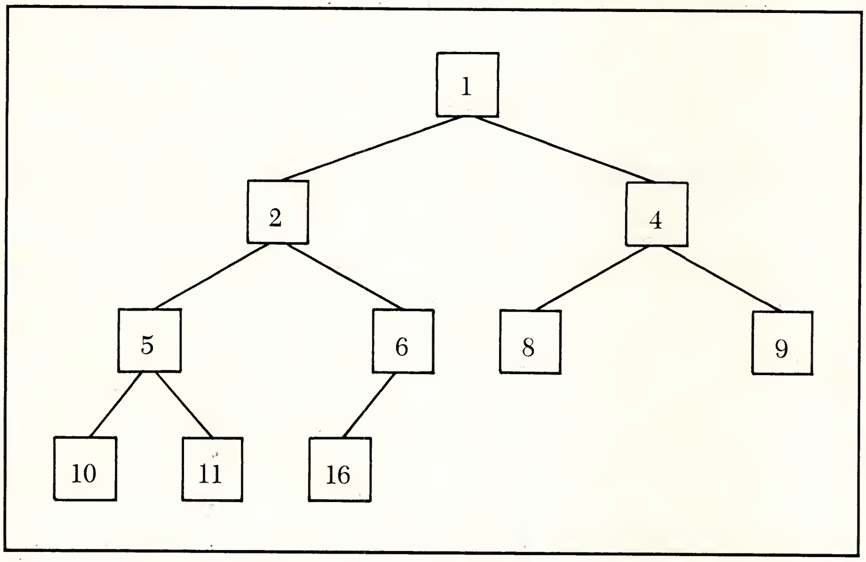


Figure 3. The binary tree produced after a heap sort.

when the array is finally sorted. (Everything else may be wrong, but we got at least one right.)

The process is then repeated on the portion of the array which comes before the original pivot, and on the portion which follows the original pivot. And so on and so on, until we reach list portions which are one element long. (These need no further sorting.)

The selection sort

This sort (line 55000) is a model of simplicity. One merely searches through the unsorted part of the list (initially the whole list) to find the largest element and the location where it occurs. You then swap this element with the element in the last position of the unsorted part of the list.

Now decrease the length of the unsorted portion by one and repeat the process until the unsorted portion has only one thing on it.

What could be easier?

The Shell sort

The Shell sort is named for its originator, Robert Louis Shell. The implementation given here (line 60000) can be thought of as a bubble sort that has a little sense.

The main problem with the bubble sort is that when two elements are swapped they are right next to one another. So, if a given element is very far away from where it ought to be, then it's going to take a great many swaps to move it into the correct position.

The Shell sort tries to get around this problem by making at least some comparisons between elements which are far apart. Then when a swap takes place, a great deal of progress may be made.

We begin by comparing elements which are N/2 positions apart and swapping them if they are out of order. Passes comparing elements N/2 units apart continue until a pass is made where no swaps are needed. Following that, passes are made comparing elements N/4 units apart. Then N/8 units. Then N/16. And so on.

Because of this process of breaking up the list, the Shell sort is often described as working on a collection of interlaced arrays all living inside a master array. In any event, the process continues until the final passes are made comparing adjacent elements.

At that stage, the Shell sort behaves exactly like the bubble sort—the hope is that by the time we begin to compare next door neighbors, most elements will be very close to their final positions. (I must note that this version of the Shell sort is not the only possible one—it is not even the most efficient—but it seems as simple as any.)

When to use what sort?

The question to be addressed in this section is a very difficult one. In addition to the space-versus-time tradeoff which is so familiar in computing, there is also the consideration of just how much programmer effort one can afford to invest in coding and debugging in order to achieve savings at execution time. Nonetheless, some general guidelines can be given.

By looking at the average-case behavior as described in Table 1, it is clear that our seven sorts divide rather neatly into three categories.

There are the three "simple" sorts—bubble, insertion, and selection—each of which has a "quadratic" run time: it increases in proportion to the square of the length of the list to be sorted.

At the other end of the spectrum are heap, merge, and quick—three "sophisticated" sorts with an expected run time proportional to nlog(n), where n is the length of the list.

(As an aside, it can be shown that nlog(n) is as fast as any sort can possibly run, provided that the sort operates by comparing pairs of list elements. That's not necessarily true of sorts in general. The radix sort mentioned above is faster and can actually get more efficient as it handles more and more records. There are complicating factors, of course. Again, see Lutowksi's article and the letters columns in the following two issues.)

Then there is the Shell sort, whose theoretical analysis is far from complete

but which still clearly occupies a middle ground.

The simple sorts should only be considered when the number of items to be sorted is known to be very small—no more than a couple of hundred—or when the application is run very infrequently and time is not a real factor.

A glance at the timing results in Table 2 reveals that the insertion sort is the clear winner among the simple sorts. (All of these have a run time proportional to n^2, but the constant of proportionality is small for the insertion sort and larger for the others.) Indeed, if you know in advance that the list of things to be sorted is very small (say, a dozen or so), the insertion sort will outrun any of the others.

The Shell sort has much to recommend it. It is relatively short—easy to code and debug—and provides a great improvement over the simple sorts for all but the smallest data sets. For these reasons, the Shell sort is often touted as the sort of choice for moderate sized lists (in the hundreds to the low thousands). These arguments are even more compelling if one chooses a version of the Shell sort which is more efficient than the one given in Listing 1. (See the section on "Possible improvements.")

For all large lists, one is forced to use one of the sophisticated sorts. Of these, quicksort is by far the most widely used. Table 2 makes the major reason for this popularity clear—quicksort is indeed quick. It also needs to be noted that it is the easiest of the three faster sorts to code and debug. Also, the merge sort has the disadvantage that it requires about twice as much storage as any of our other sorts—two arrays instead of one.

Size of List	Bubble	Heap	Insertion	Merge	Quick	Selection	Shell
32	12	6	5	8	4	8	11
64	57	16	23	19	11	31	27
128	218	36	88	43	26	119	114
256	869	84	333	95	57	470	268
512	3473	190	1333	210	123	1866	807
1024	13874	426	5212	463	291	7432	2671

Table 2. Relative run-time comparisons of the seven sorts based on actual runs sorting randomly ordered lists. (See Table 1 for theoretical run times.)

What sort of shape is your data in?

It must be emphasized that the discussion so far has been predicated on the assumption that the data to be sorted is roughly random. If that is not the case, it may well be possible to take advantage of the known structure of the data in selecting the sort to be used.

To take one rather far-fetched example, suppose that the input list happened to be already in the correct order. In such a case, the bubble sort runs in linear time: directly proportional to the number of items. (It goes through the list once, recognizes that no swaps were made, and quits.) This is the best-case behavior of the bubble sort, and it is very good.

By way of contrast, it is in exactly this situation that quicksort displays its worst-case behavior and runs in quadratic time—much slower than the bubble sort. (There are ways around this unfortunate situation—see below.)

A more frequently occurring special data distribution is where the list to be sorted consists of a relatively large correctly ordered list followed by a few new elements which need to be put in their correct locations. Such a situation

Listing 1. **VSORTS.BAS is written for Microsoft BASIC running under CP/M on the H8 or H/Z89 or under CP/M-85 on the H/Z100. Its menu will let you choose among the seven sorts discussed in the article; then you can select the number of elements to be sorted (from one to 50) and whether the list to be sorted will be ordered, reverse ordered, or random. VSORTS will sort the elements and give you a visual representation of the sorting process.**

```
1000 '  VSORTS.BAS -    A program to provide visual displays of seven
                        sorts using Heath/Zenith graphics

1100 '                  Bob Crawford
                        May 1984

2000 '  Main program

2100 GOSUB 4000                  'Initialize
2200 WHILE TRUE
2300    GOSUB 7000               'Display menu
2400    IF CHOICE = 8
        THEN 63000               'Exit the program
2500    GOSUB 9000               'Shuffle and display the list
2600    ON CHOICE GOSUB 15000,20000,30000,35000,45000,55000,60000
2700    PRINT EXGR$;
2800    PRINT FNS$(14,62); RV$; "Finished!"; NV$;
2900    IF INKEY$ = "" AND FLAG THEN 2900
3000    PRINT FNS$(25,1); EEL$; CH$; CS$
3100 WEND

4000 '  Initial set-up

4100 ON ERROR GOTO 63000
4200 DEFINT A-Z
4300 DIM X(50), Y(50), STACK(10,2)
4400 RANDOMIZE(PEEK(11))
4500 ESC$ = CHR$(27)             'ESCAPE
4600 CS$ = ESC$ + "E"            'CLEAR SCREEN
4700 EEL$ = ESC$ + "K"           'ERASE TO END OF LINE
4800 EN25$ = ESC$ + "x1"         'ENABLE 25TH LINE
4900 DIS25$ = ESC$ + "y1"        'DISABLE 25TH LINE
5000 CH$ = ESC$ + "H"            'CURSOR HOME
5100 CUROFF$ = ESC$ + "x5"       'CURSOR OFF
5200 CURON$ = ESC$ + "y5"        'CURSOR ON
5300 RV$ = ESC$ + "p"            'REVERSE VIDEO
5400 NV$ = ESC$ + "q"            'NORMAL VIDEO
5500 GR$ = ESC$ + "F"            'ENTER GRAPHICS MODE
5600 EXGR$ = ESC$ + "G"          'EXIT GRAPHICS MODE
5700 DEF FNS$(I,J) = ESC$+"Y"
                  +CHR$(I+31)
                  +CHR$(J+31) 'DIRECT CURSOR ADDRESSING FUNCTION
5800 DEF FNC(A$) =
          (80 - LEN(A$))/2    'CENTERING FUNCTION
5900 DEF FNN$(I,A$) =
        FNS$(I,FNC(A$)) + A$  'CENTERED PRINT AT LINE I, NORMAL VIDEO
6000 DEF FNF(X) = 26 - (X+1)\2   'HEIGHT LOCATOR FOR GRAPHICS
6100 TRUE = -1
6200 FALSE = 0
6300 PRINT CS$
6400 RETURN

7000 '  Routine to display the menu

7100 PRINT CS$
7200 PRINT FNN$(3,"Here are the choices:")
7300 PRINT FNN$(6,"1) Bubble Sort   ")
7400 PRINT FNN$(8,"2) Heap Sort     ")
7500 PRINT FNN$(10,"3) Insertion Sort")
7600 PRINT FNN$(12,"4) Merge Sort    ")
7700 PRINT FNN$(14,"5) Quick Sort    ")
7800 PRINT FNN$(16,"6) Selection Sort")
7900 PRINT FNN$(18,"7) Shell Sort    ")
8000 PRINT FNN$(20,"8) Exit          ")
8100 PRINT FNN$(23,"  Your choice?  ");
8200 INPUT "", CHOICE
8300 IF CHOICE < 1 OR CHOICE > 8
     THEN PRINT FNS$(23,1); EEL$ :
         GOTO 8100
8400 PRINT CS$
8500 RETURN

9000 '  Routine to shuffle and display the list

9100 PRINT FNN$(12,"How many elements do you want to sort (1-50)");
9200 INPUT N
9300 PRINT CS$;
9400 IF N < 1 OR N > 50 THEN 9100
9500 PRINT FNN$(7,"Please select the type of list you wish:");
9600 PRINT FNN$(10,"1. Random list        ");
```

→

can dramatically extend the utility of the insertion sort to much larger lists.

Possible improvements

Having settled on the insertion, Shell, and quick- sorts as being the best in their respective categories, we can look briefly at some modifications to these sorts which can improve their performance.

The insertion sort

Since the insertion sort is so simple, it is difficult to see how major changes could be made without changing the very nature of the sort. However, there is one change (which applies to sorting techniques in general, not just the insertion sort) which can often effect a major improvement.

Many times the items on the list are very large while the key field on which the sorting is to be done is relatively small. (A common example is a client or membership list to be sorted in ZIP-code order.) In such a case, the amount of time it takes to move a list item is far greater than the amount of time it takes to compare two keys.

In that case, it does not pay to keep the data in a simple array. A better idea is a linked list, where each record contains a field indicating the previous record and the following one.

The necessary comparisons keep the run time of the sorting method itself quadratic; but only a linear number of expensive data movements are required. Therefore, the constant of proportionality remains small.

The Shell sort

The Shell sort offers far more opportunities for improvement than the insertion sort.

As was mentioned during its description, the Shell sort may be viewed as a technique which works on a collection of interlaced subarrays all living inside the master array which is to be sorted. These subarrays are determined in a series of stages—comparing N/2 positions apart, N/4, etc.

In other words, an increment (call it Δ) is chosen for a given stage, and a typical subarray has the form X(J), X(J + Δ), X(J + 2*Δ), etc.

At any one stage of the sort, a whole bunch of subarrays are being looked at one after the other—one for each suitable value of J.

After the subarrays are sorted, a different increment Δ is chosen, and the process is repeated until the entire array is sorted. (It is easy to guarantee that the complete array does in fact end up sorted—simply make sure that the last increment value used is $\Delta = 1$.)

Thus the Shell sort offers two major areas for improvement—in how we choose the sequence of increments, and in how we do the sorting of each subarray.

Given what we have already said concerning the simple sorts, the question of

how to sort the subarrays is easy—use the insertion sort! End of discussion.

The question of the best increment choice has received considerable study with no definitive results. However, it is clear that the method used in our program (to start with $\Delta = N/2$ and divide Δ by two at each stage) is not the best in most cases.

One sequence for values of Δ which has no strong theoretical basis, but which has proven effective in practice, is:
... 9841, 3280, 1093, 364, 121, 40, 13, 4, 1.

This sequence is generated by starting with the value $X = 1$ and repeatedly replacing X by $3 * X + 1$ until you reach a value for X that is greater than N. The starting value for Δ is X/3, and successive values for Δ are obtained by dividing Δ by 3. (These are both integer divisions.) Using the insertion sort on sublists and this sequence of increments, the time for the Shell sort has been variously estimated as $n(\log n)^2$ and $n^{1.25}$.

Quicksort

The primary efforts at improving the quicksort have focussed on trying to get around the worst-case behavior. (Remember, quicksort runs in quadratic time on ordered or reverse-ordered lists.)

One simple and popular approach is to use the so-called median-of-three method for choosing the pivot point for quicksort. Recall that quicksort breaks the list into two sublists—one consisting of the things smaller than the pivot and one consisting of the things larger than the pivot—and then goes to work on those sublists.

As with most of these divide-and-conquer algorithms, things work out best when the two subproblems (our two sublists) are about the same size—roughly half the size of the original problem. By the same token, the situation is at its worst when one subproblem is tiny and the other is huge.

In effect, the divide-and-conquer algorithms try to short-circuit the N^2 run time. If they can work on two equal-sized lists, then they can get, not half the run time, but a quarter—and to the extent they can divide those sublists equally, the savings continue.

Thus, things would be optimal if the pivot chosen were to be the median of the original list. (But ask somebody how to find the median of a list and they'll probably say "sort the list and look in the middle"—that doesn't help.)

The quicksort in Listing 1 uses a simple method for finding the pivot—take the first element in the list. Simple, yes, but when the list is in order or reverse order, it results in one subproblem being to sort the empty list (pretty tiny) and the other being to sort the list consisting of everything but the pivot (pretty huge)—thus the terrible runtime.

```
9700 PRINT FNN$(12,"2. Ordered list       ");
9800 PRINT FNN$(14,"3. Reverse ordered list.");
9900 PRINT FNN$(17,"Your choice? ");
10000 INPUT "", LISTTYPE
10100 IF LISTTYPE < 1 OR LISTTYPE > 3
     THEN PRINT FNS$(17,1); EEL$ :
         GOTO 9900
10200 PRINT CS$;
10300 PRINT CUROFF$; GR$; EN25$;
10400 IF LISTTYPE <> 3 THEN 10900
10500 FOR I = 1 TO N
10600    X(I) = N - I + 1
10700 NEXT I
10800 GOTO 11700
10900 FOR I = 1 TO N
11000    X(I) = I
11100 NEXT I
11200 IF LISTTYPE = 2 THEN 11700
11300 FOR I = 1 TO N
11400    J = INT(N*RND + 1)
11500    SWAP X(I), X(J)
11600 NEXT I
11700 FOR I = 1 TO N
11800    PRINT CH$
11900    PRINT FNS$(FNF(X(I)),I);
12000    IF X(I) MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
12100 NEXT I
12200 PRINT CH$
12300 PRINT EXGR$; FNS$(5,60);
12400 PRINT "Press any key";
12500 PRINT FNS$(7,62); "to begin";
12600 PRINT FNS$(9,62); "the sort";
12700 IF INKEY$ = "" THEN 12700
12800 PRINT FNS$(5,60); EEL$;
12900 PRINT FNS$(7,62); EEL$;
13000 PRINT FNS$(9,62); EEL$;
13100 RETURN

14000 ' Prompt routine

14100 PRINT FNS$(8,60); "Press any key";
14200 PRINT FNS$(10,63); "to stop";
14300 PRINT FNS$(12,63); "the sort";
14400 PRINT GR$
14500 RETURN

15000 ' Bubble sort routine -- shifts display as it goes

15100 FLAG = TRUE
15200 PRINT FNS$(5,61); RV$; "Bubble Sort"; NV$
15300 GOSUB 14000
15400 TOP = N-1
15500 LAST.SWAP = 3
15600 WHILE LAST.SWAP > 2
15700    LAST.SWAP = 0
15800    FOR I = 1 TO TOP
15900       IF X(I+1) >= X(I) THEN 16900
16000       PRINT FNS$(FNF(X(I)),I); " ";
16100       PRINT FNS$(FNF(X(I+1)),I+1); " ";
16200       PRINT FNS$(FNF(X(I+1)),I);
16300       IF X(I+1) MOD 2 = 1
            THEN PRINT "m";
            ELSE PRINT "n";
16400       PRINT FNS$(FNF(X(I)),I+1);
16500       IF X(I) MOD 2 = 1
            THEN PRINT "m";
            ELSE PRINT "n";
16600       SWAP X(I),X(I+1)
16700       PRINT CH$
16800       LAST.SWAP = I+1
16900    NEXT I
17000    TOP = LAST.SWAP - 1
17100    IF INKEY$ <> ""
        THEN FLAG = FALSE :
        RETURN
17200 WEND
17300 PRINT CH$
17400 RETURN

20000 ' Heapsort routine -- shifts display as it goes

20100 FLAG = TRUE
20200 PRINT FNS$(5,62); RV$; "Heap Sort"; NV$
20300 GOSUB 14000              'Display the prompt
20400 FOR K = 2 TO N
```

```
20500    I = K
20600    Y = X(K)
20700    J = INT(I/2)
20800    WHILE J>0
20900       IF Y <= X(J) THEN 21800
21000       PRINT FNS$(FNF(X(I)),I); " ";
21100       PRINT FNS$(FNF(X(J)),I);
21200       IF X(J) MOD 2 = 1
           THEN PRINT "m";
           ELSE PRINT "n";
21300       PRINT CH$
21400       X(I) = X(J)
21500       I = J
21600       J = INT(I/2)
21700    WEND
21800    PRINT FNS$(FNF(X(I)),I); " ";
21900    PRINT FNS$(FNF(Y),I);
22000    IF Y MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
22100    PRINT CH$
22200    X(I) = Y
22300 NEXT K
22400 FOR K = N TO 2 STEP -1
22500    Y = X(K)
22600    PRINT FNS$(FNF(X(K)),K); " ";
22700    PRINT FNS$(FNF(X(1)),K);
22800    IF X(1) MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
22900    PRINT CH$
23000    X(K) = X(1)
23100    I = 1
23200    J = 2
23300    IF (X(3) > X(2)) AND (K-1 >= 3)
        THEN J = 3
23400    WHILE 1 = 1
23500       IF J > K-1 THEN 24700
23600       IF X(J) <= Y THEN 24700
23700       IF X(J) <= Y THEN 24700
23800       PRINT FNS$(FNF(X(I)),I); " ";
23900       PRINT FNS$(FNF(X(J)),I);
24000       IF X(J) MOD 2 = 1
           THEN PRINT "m";
           ELSE PRINT "n";
24100       PRINT CH$
24200       X(I) = X(J)
24300       I = J
24400       J = 2*I
24500       IF J+1 <= K-1
            THEN IF X(J+1) > X(J)
            THEN J = J+1
24600    WEND
24700    PRINT FNS$(FNF(X(I)),I); " ";
24800    PRINT FNS$(FNF(Y),I);
24900    IF Y MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
25000    PRINT CH$
25100    X(I) = Y
25200 IF INKEY$ <> ""
     THEN FLAG = FALSE :
          RETURN
25300 NEXT K
25400 RETURN

30000 ' Straight insertion sort routine -- shifts display as it goes

30100 FLAG = TRUE
30200 PRINT FNS$(5,59); RV$; "Insertion Sort"; NV$
30300 GOSUB 14000               'Display the prompt
30400 FOR J = 2 TO N
30500    I = J-1
30600    S = X(J)
30700    WHILE I>0 AND S < X(I)
30800       PRINT FNS$(FNF(X(I+1)),I+1); " ";
30900       PRINT FNS$(FNF(X(I)),I+1);
31000       IF X(I) MOD 2 = 1
           THEN PRINT "m";
           ELSE PRINT "n";
31100       PRINT CH$
31200       X(I+1) = X(I)
31300       I = I-1
31400    WEND
31500    PRINT FNS$(FNF(X(I+1)),I+1); " ";
31600    PRINT FNS$(FNF(S),I+1);
```

→

The median-of-three method tries to make a reasonably intelligent choice for the pivot while not incurring any great computational expense. What it does is to look at the first element in the list, the last element in the list, and the middle element in the list and take the pivot to be the median of those three. It is indeed worth the effort if there is any reasonable chance that your list will be ordered (or nearly so).

Another recent development, again aimed at obtaining a better choice for the pivot, is called meansort.

On the first pass through the list, meansort behaves just like our version of quicksort. Along the way, though, it collects statistics on the list and thereafter uses the mean value of each sublist as the pivot for that sublist. For many types of distributions, this results in a fair sized improvement. Its primary advantage, however, would seem to be that the sort time is independent of the order of the list.

To end our discussion of improvements, it should be mentioned that many sophisticated sort routines will use a combination of methods.

The advanced algorithms like quicksort and heapsort, which behave so well on large lists, are quite inefficient on short lists; there, something like the insertion sort will outperform them. Thus, for example, it is not uncommon for a sort to begin by acting like quicksort only to switch over to acting like the insertion sort when the sublists to be sorted reach some critical short length. Such an approach will often be better than any of the individual methods used alone.

What to expect when you run the program

Now that we've covered the theoretical issues, perhaps you'd like to experiment with sorting. The program presented here will allow you to watch each of the seven sorts in action on lists with characteristics of your choice.

The program VSORTS.BAS (standing for visual sorts) is menu-driven. Initially you will see the main menu indicating the choices you have:

1) Bubble Sort
2) Heap Sort
3) Insertion Sort
4) Merge Sort
5) Quick Sort
6) Selection Sort
7) Shell Sort
8) Exit

You will then be asked for the number of elements you want to sort, which can be anywhere from 1 to 50. Finally, you must indicate the type of list you wish to work on (random, ordered, or reverse ordered).

When you have made all the choices, the initial list of the size and type you chose will appear as a collection of small

```
31700    IF S MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
31800    PRINT CH$
31900    X(I+1) = S
32000    IF INKEY$ <> ""
        THEN FLAG = FALSE :
        RETURN
32100 NEXT J
32200 PRINT CH$
32300 RETURN

35000 ' Straight merge sort routine -- shifts display as it goes

35100 FLAG = TRUE
35200 PRINT FNS$(5,61); RV$; "Merge Sort"; NV$
35300 GOSUB 14000                'Display the prompt
35400 DELTA = 1
35500 WHILE DELTA < N
35600    START1 = 1
35700    START2 = START1 + DELTA
35800    WHILE START2 <= N
35900       STOP1 = START2 - 1
36000       STOP2 = START2 + DELTA - 1
36100       IF STOP2 > N THEN STOP2 = N

36200 '    Merge the lists X(START1),...,X(STOP2) and
           X(START2),...,X(STOP2) into Y(START1),...,Y(STOP2)

36300       MARK1 = START1
36400       MARK2 = START2
36500       YMARK = START1
36600       WHILE MARK1 <= STOP1 AND MARK2 <= STOP2
36700          IF X(MARK1) < X(MARK2) THEN 36800 ELSE 37500
36800             PRINT FNS$(FNF(X(YMARK)),YMARK); " ";
36900             PRINT FNS$(FNF(X(MARK1)),YMARK);
37000             IF X(MARK1) MOD 2 = 1
                 THEN PRINT "m";
                 ELSE PRINT "n";
37100             PRINT CH$
37200             Y(YMARK) = X(MARK1)
37300             MARK1 = MARK1 + 1
37400          GOTO 38100
37500             PRINT FNS$(FNF(X(YMARK)),YMARK); " ";
37600             PRINT FNS$(FNF(X(MARK2)),YMARK);
37700             IF X(MARK2) MOD 2 = 1
                 THEN PRINT "m";
                 ELSE PRINT "n";
37800             PRINT CH$
37900             Y(YMARK) = X(MARK2)
38000             MARK2 = MARK2 + 1
38100          YMARK = YMARK + 1
38200       WEND
38300       FOR I = MARK1 TO STOP1
38400          PRINT FNS$(FNF(X(YMARK)),YMARK); " ";
38500          PRINT FNS$(FNF(X(I)),YMARK);
38600          IF X(I) MOD 2 = 1
              THEN PRINT "m";
              ELSE PRINT "n";
38700          PRINT CH$
38800          Y(YMARK) = X(I)
38900          YMARK = YMARK + 1
39000       NEXT I
39100       FOR I = MARK2 TO STOP2
39200          PRINT FNS$(FNF(X(YMARK)),YMARK); " ";
39300          PRINT FNS$(FNF(X(I)),YMARK);
39400          IF X(I) MOD 2 = 1
              THEN PRINT "m";
              ELSE PRINT "n";
39500          PRINT CH$
39600          Y(YMARK) = X(I)
39700          YMARK = YMARK + 1
39800       NEXT I

39900       IF INKEY$ <> ""
           THEN FLAG = FALSE :
                RETURN
40000 '     Copy the Y-list back into the X-list

40100       FOR I = START1 TO STOP2
40200          X(I) = Y(I)
40300       NEXT I

40400       START1 = STOP2 + 1
40500       START2 = START1 + DELTA
40600    WEND
```

```
40700    DELTA = 2*DELTA
40800 WEND
40900 RETURN

45000 ' Quicksort routine -- shifts display as it goes

45100 FLAG = TRUE
45200 PRINT FNS$(5,61); RV$; "Quick Sort"; NV$
45300 GOSUB 14000                   'Display the prompt
45400 STACK(1,1) = 1
45500 STACK(1,2) = N
45600 STACKTOP = 1
45700 WHILE STACKTOP > 0
45800    UPPER.BOUND = STACK(STACKTOP,2)
45900    LOWER.BOUND = STACK(STACKTOP,1)
46000    STACKTOP = STACKTOP - 1
46100    WHILE UPPER.BOUND > LOWER.BOUND
46200       A = X(LOWER.BOUND)
46300       J = LOWER.BOUND
46400       UP = UPPER.BOUND
46500       DOWN = LOWER.BOUND
46600       WHILE 1=1
46700          WHILE (UP > DOWN) AND (X(UP) >= A)
46800             UP = UP - 1
46900          WEND
47000          J = UP
47100          IF UP = DOWN THEN 48800
47200          PRINT FNS$(FNF(X(DOWN)),DOWN); " ";
47300          PRINT FNS$(FNF(X(UP)),DOWN);
47400          IF X(UP) MOD 2 = 1
               THEN PRINT "m";
               ELSE PRINT "n";
47500          PRINT CH$
47600          X(DOWN) = X(UP)
47700          WHILE (DOWN < UP) AND (X(DOWN) <= A)
47800             DOWN = DOWN + 1
47900          WEND
48000          J = DOWN
48100          IF DOWN = UP THEN 48800
48200          PRINT FNS$(FNF(X(UP)),UP); " ";
48300          PRINT FNS$(FNF(X(DOWN)),UP);
48400          IF X(DOWN) MOD 2 = 1
               THEN PRINT "m";
               ELSE PRINT "n";
48500          PRINT CH$
48600          X(UP) = X(DOWN)
48700       WEND
48800       PRINT FNS$(FNF(X(J)),J); " ";
48900       PRINT FNS$(FNF(A),J);
49000       IF A MOD 2 = 1
            THEN PRINT "m";
            ELSE PRINT "n";
49100       PRINT CH$
49200       X(J) = A
49300       IF J - LOWER.BOUND > UPPER.BOUND - J
            THEN 49400
            ELSE 50200
49400          I = UPPER.BOUND
49500          UPPER.BOUND = J - 1
49600          STACKTOP = STACKTOP + 1
49700          STACK(STACKTOP,1) = LOWER.BOUND
49800          STACK(STACKTOP,2) = UPPER.BOUND
49900          LOWER.BOUND = J + 1
50000          UPPER.BOUND = I
50100       GOTO 51000
50200          I = LOWER.BOUND
50300          LOWER.BOUND = J + 1
50400          STACKTOP = STACKTOP + 1
50500          STACK(STACKTOP,1) = LOWER.BOUND
50600          STACK(STACKTOP,2) = UPPER.BOUND
50700          LOWER.BOUND = I
50800          UPPER.BOUND = J - 1
50900          IF INKEY$ <> ""
               THEN FLAG = FALSE :
                    RETURN
51000    WEND
51100 WEND
51200 RETURN

55000 ' Straight selection sort routine -- shifts display as it goes

55100 FLAG = TRUE
55200 PRINT FNS$(5,59); RV$; "Selection Sort"; NV$
55300 GOSUB 14000                   'Display the prompt
55400 FOR J = N TO 2 STEP -1
55500    MAX = X(J)
```

→

Index to Advertisers

* Contact advertiser directly.

Name ____________________

Address ____________________

City ____________________

State ____________ ZIP ____________

For further information from specific advertisers, refer to index on page 82 and circle numbers below.

100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

Reader Service Card
March-April 1986

Expires June 15, 1986

Please check *only* one box for each of the following questions.

Which microcomputer system do you own/use?

K. ☐ H88 O. ☐ H/Z100-110-120 S. ☐ Z171
L. ☐ H/Z89-90 P. ☐ H/Z150-158-160 T. ☐ Z200
M. ☐ H8 Q. ☐ Z138
N. ☐ H11 R. ☐ Z148

How did you obtain this copy of *Sextant*?

1. ☐ Subscription 3. ☐ Friend 5. ☐ Other
2. ☐ Retail Store 4. ☐ Single Copy Mail Order

Please rate the articles in this issue for interest and helpfulness, using the key on page 83.

1	2	3	4	5	6	7	8	9	
☐	☐	☐	☐	☐	☐	☐	☐	☐	Excellent
☐	☐	☐	☐	☐	☐	☐	☐	☐	Good
☐	☐	☐	☐	☐	☐	☐	☐	☐	Fair
☐	☐	☐	☐	☐	☐	☐	☐	☐	Poor
☐	☐	☐	☐	☐	☐	☐	☐	☐	Didn't read

Comments: ____________________

82a1

Name ____________________

Address ____________________

City ____________________

State ____________ ZIP ____________

For further information from specific advertisers, refer to index on page 82 and circle numbers below.

100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259
260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287	288	289
290	291	292	293	294	295	296	297	298	299

Reader Service Card
March-April 1986

Expires June 15, 1986

Please check *only* one box for each of the following questions.

Which microcomputer system do you own/use?

K. ☐ H88 O. ☐ H/Z100-110-120 S. ☐ Z171
L. ☐ H/Z89-90 P. ☐ H/Z150-158-160 T. ☐ Z200
M. ☐ H8 Q. ☐ Z138
N. ☐ H11 R. ☐ Z148

How did you obtain this copy of *Sextant*?

1. ☐ Subscription 3. ☐ Friend 5. ☐ Other
2. ☐ Retail Store 4. ☐ Single Copy Mail Order

Please rate the articles in this issue for interest and helpfulness, using the key on page 83.

1	2	3	4	5	6	7	8	9	
☐	☐	☐	☐	☐	☐	☐	☐	☐	Excellent
☐	☐	☐	☐	☐	☐	☐	☐	☐	Good
☐	☐	☐	☐	☐	☐	☐	☐	☐	Fair
☐	☐	☐	☐	☐	☐	☐	☐	☐	Poor
☐	☐	☐	☐	☐	☐	☐	☐	☐	Didn't read

Comments: ____________________

82a2

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

Sextant Reader Service
P.O. Box 13558
Philadelphia, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

Sextant Reader Service
P.O. Box 13558
Philadelphia, PA 19101

```
55600    MAXSPOT = J
55700    FOR I = J-1 TO 1 STEP -1
55800       IF X(I) <= MAX THEN 56100
55900       MAX = X(I)
56000       MAXSPOT = I
56100    NEXT I
56200    PRINT FNS$(FNF(X(J)),J); " ";
56300    PRINT FNS$(FNF(X(MAXSPOT)),MAXSPOT); " ";
56400    PRINT FNS$(FNF(X(MAXSPOT)),J);
56500    IF X(MAXSPOT) MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
56600    PRINT CH$
56700    PRINT FNS$(FNF(X(J)),MAXSPOT);
56800    IF X(J) MOD 2 = 1
        THEN PRINT "m";
        ELSE PRINT "n";
56900    PRINT CH$
57000    SWAP X(J),X(MAXSPOT)
57100    IF INKEY$ <> ""
        THEN FLAG = FALSE :
             RETURN
57200 NEXT J
57300 RETURN

60000 ' Shell sort routine -- shifts display as it goes

60100 FLAG = TRUE
60200 PRINT FNS$(5,61); RV$; "Shell Sort"; NV$
60300 GOSUB 14000               'Display the prompt
60400 DELTA = N
60500 WHILE DELTA > 1
60600    DELTA = DELTA/2
60700    SWITCH = 1
60800    WHILE SWITCH = 1
60900       SWITCH = 0
61000       MAX = N - DELTA
61100       FOR I = 1 TO MAX
61200          IF X(I) <= X(I+DELTA) THEN 62300
61300          PRINT FNS$(FNF(X(I)),I); " ";
61400          PRINT FNS$(FNF(X(I+DELTA)),I+DELTA); " ";
61500          PRINT FNS$(FNF(X(I+DELTA)),I);
61600          IF X(I+DELTA) MOD 2 = 1
              THEN PRINT "m";
              ELSE PRINT "n";
61700          PRINT CH$
61800          PRINT FNS$(FNF(X(I)),I+DELTA);
61900          IF X(I) MOD 2 = 1
              THEN PRINT "m";
              ELSE PRINT "n";
62000          PRINT CH$
62100          SWAP X(I),X(I+DELTA)
62200          SWITCH = 1
62300       NEXT I
62400       IF INKEY$ <> ""
           THEN FLAG = FALSE :
                RETURN
62500    WEND
62600 WEND
62700 RETURN

63000 ' Exit routine

63100 PRINT CS$; DIS25$; EXGR$; CURON$
63200 END
```

squares. (Each square is as close to a pixel as you can get with the H/Z89's chunky graphics.) Toward the right of the screen will be a prompt asking you to press any key to begin the sort.

Each small square represents a list element. A square's horizontal position on screen indicates the subscript of the list element; the vertical position of the square indicates the value of that list element. Thus when the list is in order, you will see a line of squares extending from the lower left corner to the upper right corner of the sort display.

After you press any key in order to get the sort started, the name of the sort being used will appear highlighted. While the sort is in progress, pressing any key will return you to the main menu at the end of the next pass of that sort.

If you let the sort run to completion, the message "Finished!" will appear highlighted. Then pressing any key returns you to the main menu. Enjoy.

Suggestions for further reading

The topic of sorting is of such importance that it has an enormous literature. In general, almost any book devoted to data structures will contain a chapter or two devoted to this subject, and many of these are very good.

The standard reference on sorting is the third volume of Donald Knuth's *The Art of Computer Programming*, entitled *Sorting and Searching* (Addison-Wesley, 1973). As with the other two volumes of *The Art*, it is a monumental storehouse of information.

A relatively recent book which is an excellent source is by Robert Sedgewick and is entitled simply *Algorithms*. This contains 40 delightful chapters dealing with many of the most important algorithms in all branches of computer science. It is current, extremely well written, and has fully six chapters devoted to sorting.

Information on meansort can be found in an article by that name by Dalia Motzkin in the Association for Computing Machinery's *Communications of the ACM*, Volume 26, Number 4, April 1983, pp. 250-251. △

Please refer to the key to the right when completing the article evaluation form on the attached Reader Service Card above. By letting us know how interesting and/or helpful you found each feature in this issue of *Sextant*, you will help us choose the best articles for you in the future.

Z100 Notebook

Alison Phillips

Greetings

One of the joys associated with my computer activities is that each day brings new opportunities for learning. The fields of computer applications are practically boundless; they may include such topics as business, science, mathematics, design, programming, graphics, music, medicine . . . and the list goes on.

In this issue, I would like to share some of my learning experiences associated with the C language and with the MS-DOS family of filters. I'll start by taking a look at some C products. (For a more direct look at C programming, you might want to look at Joseph Katz's column, "C Notes.")

The C bug bites

Before going further, let me introduce C properly.

Why call a language C anyway? C could appropriately derive from Cryptic or Compact, for it is a language that is cryptic, terse, and full of abbreviations. However, C is really the letter that follows B—isn't that a revelation?

Let me explain.

C was designed sometime around 1972, by Dennis Ritchie when he was working on the Unix operating system for Bell Laboratories. But many of C's features stem from an older language called BCPL (Basic Combined Programming Language). The influence of BCPL on C proceeded indirectly through a language called B. Hence, the innocent and cryptic title C, which follows B.

All along, I have been a BASIC enthusiast and loyalist. I don't believe that it is carved in stone that one must write spaghetti code in BASIC. I sometimes think that some of those who denigrate BASIC haven't exploited the latest versions of this powerful high-level language.

But, having said that, I must tell you that I have been bitten by the C bug.

I couldn't stand it any longer—the way my learned friends patronized me when they realized that I was just a BASIC programmer. The C fever came on slowly. I called the doctor, and he said the only thing to do was to get a C compiler—an over-the-counter one would do—and to take a small dose of C each night.

I did, but the fever worsened.

Without bothering to count the cost—I suppose I thought the insurance would take care of it—I acquired, not one, but two C compilers. (I will talk about each shortly.)

Putting C into perspective

Let's put the C language into perspective and examine why it is increasing in popularity. C is a language for the serious programmer; it is cryptic, structured, elegant, and powerful. It is a mid-level language—higher than assembler, lower than BASIC.

C is often used in writing operating systems and utilities, although it also serves well as a general-purpose language. But aside from its power, it is fun to use, and C compilers produce machine code that is both efficient and fast.

At 14 years old, C is a young language; 18 years younger than FORTRAN and 10 years younger than BASIC. It lay in the doldrums for a number of years, but has rapidly gained in popularity since 1978, when Kernighan and Ritchie authored *The C Programming Language*, a book which has become known as the "K&R bible."

In recent years C books have proliferated, and so have compilers. There are now over 20 C compilers available for use with microcomputers. C has been implemented on a wide range of computers—all the way from Z80 machines to the Cray-1 supercomputer.

C code is highly transportable. And because of C's compactness and elegance, a C compiler can be written for a new microcomputer faster and more easily than compilers or interpreters for other languages. C compilers vary greatly in power and quality. With a good compiler, the difference in speed between C and assembly-language programs will go unnoticed, except in demanding cases.

In contrast with the BASIC compiler, the C compiler produces executable programs that run on their own and do not require a run-time module. (It is true that some versions of compiled BASIC produce executable programs which stand alone, but such programs are apt to be larger than equivalent C programs.)

Selecting a compiler

We have a wide selection of C compilers from which to choose. C compilers are available for the Heath Disk Operating System (HDOS), CP/M-80, CP/M-86, for the Zenith Disk Operating System (Z-DOS), and for version 2 of the Microsoft Disk Operating System (MS-DOS).

I felt that my compiler, among other things, must meet three basic requirements: (1) run under MS-DOS version 2; (2) support the full K&R imple-

mentation of C; and (3) be in the low or intermediate price range.

I first purchased the DeSmet C Compiler Development Package for $194. Later, when Toolworks C was announced, I acquired it, along with the Toolworks MATHPAK for a total of $79.90. I hasten to point out the combined cost of these two respected compilers is less than the cost of a single "big name" compiler; for example, Computer Innovations' C-86 compiler goes for $395.

The least expensive choice is Small-C. It is available in the public domain for free, but it is best to order it directly from its author, J. E. Hendricks. In this way, you can lend your support to the author and be assured of getting the latest version. It sells for $25 and the *Small-C Handbook* sells for $17.95. The cost for commercial C compilers ranges from a low of $49 to over $500.

The DeSmet C Development Package

I had been thinking about C for a long time. Before buying, I read again *Byte* magazine's C-language issue from August 1983. In that issue, Ralph A. Phraner compared nine C compilers on the MS-DOS version 1 (PC-DOS) operating system. He found the extent and quality of DeSmet C, version 1.5, to be "amazing," and that DeSmet C compiled fast and produced tight code. It ranked high in execution speed. Local users also speak well of DeSmet C.

The DeSmet C Development Package (hereafter, just DeSmet C) is marketed by the C Ware Corporation of Sunnyvale, California. It is sold unbundled. The compiler lists for $109, the Symbolic Debugger for $50, and the support for MS-DOS's LINK program for $35. This brings the total price of the package to $194, which puts it in the intermediate price range.

DeSmet C provides the following C programming environment:

(1) A two-pass compiler that reads C source code and produces object modules that may be linked together to produce an executable program.

(2) An assembler that reads assembly-language source code and produces linkable object modules.

(3) A "binder" that links together the object and library modules to get an executable program.

(4) A library program that combines object modules into library modules.

(5) A list program that produces a listing together with cross references.

(6) A profiler program that provides a statistical measure of the amount of time spent in a program or in a procedure within the program.

(7) A full-screen text editor that produces ordinary ASCII files.

(8) A full-screen debugger to execute the program step by step and display the C source lines while doing so. Breakpoints are address or line numbers.

(9) A library of run-time routines. Routines are available both for systems with and without the 8087 math coprocessor.

(10) A LINK support program. Both the assembler and compiler produce object modules that are different from those produced by Microsoft's assembler, MASM. You'll need the LINK support if you wish to produce object modules in Microsoft .OBJ format, or if you wish to link modules produced by MASM.

The Toolworks C compiler package

Toolworks C has a good heritage. It is based upon the highly regarded Toolworks C/80, a CP/M compiler used with eight-bit machines such as the H/Z89. C/80 was ranked high in execution speed and efficiency in the January '83 issue of *Byte*.

Toolworks C supports all versions of the MS-DOS operating system. It runs on all H/Z100 and H/Z150 series computers, as well as on the IBM Personal Computer. It supports a virtually complete subset of the C programming language as defined by Kernighan and Ritchie in *The C Programming Language*.

The Software Toolworks, long known for its affordable software, has priced Toolworks C very competitively. The price of Toolworks C is $49.95 and the optional MATHPAK is $29.95. For the

budget minded, it is worth noting that the complete Toolworks C package costs less than half as much as the DeSmet C package.

MATHPAK adds 32-bit floating-point and signed integer math capability to Toolworks C. MATHPAK supports trigonometric, hyperbolic, and logarithmic functions. You can start with the integer-only compiler (Toolworks C) and add the MATHPAK at a later date if desired. The MATHPAK is not a subroutine library or an add-on. Rather, when installed, it becomes fully integrated with the Toolworks C compiler.

The complete package provides the following C programming environment:

(1) A two-pass compiler that reads C source code and produces linkable object code.

(2) A choice of library files for MS-DOS 1.x (Z-DOS) or MS-DOS 2.x, or both. The library files are about twice as large as those provided with DeSmet C.

(3) The source files for the entire library—about 316 kilobytes worth.

(4) A total of 11 source-file examples, which include a Tree Sort and the Sieve of Eratosthenes.

DeSmet C documentation

The DeSmet C Development Package manual contains about 144 pages. The manual has 13 sections and five appendices. Page numbering starts over with each section, and each section stands alone. It brings to mind the early Digital Research CP/M documentation of ill fame. The manual contains no index, but it somewhat compensates for this deficiency by providing a detailed table of contents.

The coverage provided by the manual is extensive, since it covers a complete and unique development package. But the coverage of each item of the package is thin.

In addition to documentation for the compiler per se, there are sections of the manual devoted to the assembler, binder, library creator, and editor. Each of these is unique to DeSmet C. Having unique elements places a burden on the manual's authors to document them; it also places a burden on the user to read and learn them. (Toolworks C avoided the problem by providing for the use of existing MS-DOS programs such as MASM, LINK, and BSE.)

I cannot say that the DeSmet manual is at all helpful to the beginner. It is not friendly, and it leaves too much unsaid. The explanations provided are as sparse as C itself.

An example of the sparseness of the documentation is the discussion of "Using the Compiler." Only one page is devoted to this section—and it includes explanations of seven supporting switches.

The entire explanation of what the compiler does and how it accomplishes its work consists of just one line.

That degree of coverage may be acceptable to professional C programmers, or to those who have lost their curiosity along the way, but it leaves me unsatisfied.

Toolworks C documentation

The Toolworks C compiler manual contains 73 pages; the MATHPAK manual provides another 23 pages. In general, each manual is carefully structured and well done.

The compiler manual contains an index with about 250 entries. The index is helpful and welcome as it stands, but another 25 or so entries are needed to fill gaps and provide better cross referencing.

I would say that the manual is friendly, but it is definitely written for an experienced computerist who has considerable understanding of MS-DOS and of programming. As the manual itself states, it is not a tutorial to introduce C, and so an excellent bibliography has been provided.

It is, however, rather unkind for the Toolworks to refer the new user, without caveat, to *The C Programming Language* by Kernighan and Ritchie. This is a stiff book, written in part by the designer of the C language, and it is written for professional programmers. There is a wide gap between it and the Toolworks C manual. It would be a challenge indeed to learn C from just these two sources.

It seems to me that all providers of low-cost, entry-level C compilers should recognize that the majority of their patrons are likely to be first-time users, not professional programmers. Accordingly, they should provide half-again more explanation to ease the transition into the language.

Toolworks C has gone a long way to help the beginner by providing a good readable manual and by providing the source code to their entire library of functions. A few more pages telling how to extract selected source code from the library and how to use the library as a learning tool would be of great value to the first-time user of the compiler.

First impressions

It is dangerous to form opinions about people or pieces of software until you have lived with them for a while. Nevertheless, first impressions are important—and some things about the two C packages are apparent, even at first glance.

Caution aside, here are my initial observations.

The Toolworks C and DeSmet C compiler packages are not much alike. It is easier to contrast them than compare them.

To start with—and to end with, in many cases—there is the matter of price. The cost of the complete Toolworks package is less than half of the cost of the DeSmet package. The fact that The Software Toolworks chose to use Microsoft's LINK, LIB, MASM, and optionally BSE, to support its compiler, and not reinvent the wheel, no doubt helped in holding down the cost.

The Toolworks C compiler produces relocatable object modules that can be directly linked using MS-DOS's LINK program. This suits me just fine.

Microsoft's LINK program has withstood years of testing. Moreover, it is "free," since it is furnished as part of the MS-DOS operating-system environment. On the other hand, DeSmet C uses a "binder" program to bind its object modules and library files into an executable program. You must pay $35 extra for the LINK support if, for instance, you wish to produce object modules in a form that LINK will accept.

The Toolworks C compiler does not have an editor integral to it. DeSmet C does. This is conceivably a plus for DeSmet, if one chooses to learn a new editor and plans to use it extensively. For me, it is a minus. I'd rather use my own editor, with which I am familiar, than pay for and learn yet another one.

DeSmet C provides a profiler program, which generates statistical measures of the time spent on program procedures. Toolworks C does not have a separate profiler program; but you can use its -P switch at compile time. This will provide statistical measures to profile your routines.

DeSmet C provides a comprehensive full-screen debugger program in its package. This debugger is closely knit with the DeSmet compiler, and it is unlikely that it would be of use with any other compiler. The unbundled cost is $50. Toolworks C does not provide a debugger; the user must rely on compil-

er error messages to find errors.

One big plus for Toolworks C is that the source code for the entire library of functions is furnished on the distribution disks. For the dedicated C programmer, this library can provide a wealth of programming information and guidance. DeSmet C doesn't provide source code for the library.

Finally, there is the matter of documentation. For the beginner, the Toolworks C manual is far better. It shows more care in its structure and in content. Also, it provides useful insights relating the compiler to the language and the operating system.

Now we will leave C for this session and turn our attention to MS-DOS.

Filters, pipes, and redirection

MS-DOS version 2 is a cut above Z-DOS because of hierarchical files, smart batch commands, filters, pipes, and redirection features. We need to talk about filters, pipes, and redirection all in one breath because they are used together. However, it is best to start with the definition of filters, and go from there.

MS-DOS has four filters; they are programs named CIPHER, FIND, MORE, and SORT.

The name "filter" aptly derives from the everyday sense of filters—devices which refine or alter the input in some way. Each filter program receives input, either from the keyboard or from another program; then it does something with the input, such as sort it, or encrypt it. The altered data is then output.

The MS-DOS term "piping" (which rhymes with typing) derives from the word "pipe"; it's not to be confused with the CP/M term "PIPping" (which rhymes with skipping).

(As I've pointed out before, the term "pipe" comes to us via the Unix operating system developed by Bell Laboratories. On the other hand, the term "PIPping" comes to us via Digital Research Incorporated, the creators of the CP/M operating sytem. PIP is an acronym for "Peripheral Interface Program." PIP means to copy from one peripheral to another.)

When an MS-DOS pipe command is given, it causes the operating sytem to take the output from one program and conduct (pipe) it to provide the input to another program. MS-DOS pipe commands work only when the programs which are piped use the standard input (STDIN) and standard output (STDOUT). Both STDIN and STDOUT are function requests, or calls, which are an intrinsic part of the MS-DOS operating system.

For piping, the operator is the vertical-bar or stile character (|). The program data is always piped from the left of the stile to the right. If you take note, all MS-DOS commands flow from left to right (unless redirected by the redirection operators).

A simple example of a pipe is: DIR |

Circle #148 on Reader Service Card

Circle #243 on Reader Service Card

MORE. Here, MORE will take the directory display and send it to the screen in pages. (Of course, you may accomplish the same results by using the DIR /P command, where the /P switch causes the display to be paged.)

Before proceeding, please recall that MS-DOS assigns the .$$$ extension to temporary work files. The DIR | MORE command works in this way: MS-DOS creates a temporary file by the name of %PIPE1.$$$. The file %PIPE1.$$$ then receives the piped output from the DIR command and holds it for use of the MORE program. The MORE program then reads the %PIPE1.$$$ file and counts the lines until the screen is almost full; then it displays "-- More --", and waits for your carriage return before continuing with the next page. After the directory has been displayed, MS-DOS erases the temporary file.

Now we will have a look at redirection. The same operation that MS-DOS accomplished using piping and the DIR | MORE command can be accomplished using redirection. DIR | MORE is equivalent to:

```
DIR > %PIPE1.$$$
MORE < %PIPE1.$$$
ERASE %PIPE1.$$$
```

When we say redirection we are referring to input/output (I/O) redirection. We use the operators < and > to show the direction of data flow between files or devices.

Formally, redirection takes place when either we redefine STDIN so that the system obtains input from a source other than the default STDIN (keyboard), or we redefine STDOUT so that the system sends output to a destination other than the default STDOUT (screen).

Suppose we want to make a permanent record of a large directory, and we want to group all of the .COM files together and all of the .EXE files together. Redirection handles this task nicely when the following commands are issued sequentially from the command line:

```
DIR *.COM > DIRECTRY.FIL
DIR *.EXE >> DIRECTRY.FIL
TYPE DIRECTRY.FIL > PRN
```

With the commands above, we redirected the output of DIR *.COM away from the screen and into a file named DIRECTRY.FIL. Then we appended the output of DIR *.EXE to the end of the file. Notice the use of the double redirection symbols that are required to append an output to an existing file. With the TYPE command, we redirected the output of TYPE away from the screen, where it would normally be displayed, to the printer instead.

The series of three commands shown above gives the general idea of redirection. Later, I'll give some other examples showing how we can redirect in reverse direction.

Using SORT.EXE to sort fields in a data file

Now let's see how the filter SORT.EXE will work for us. This useful program is furnished on the MS-DOS 2 distribution disk. Unfortunately, it is not a part of the Z-DOS distribution package.

SORT is a transient (or disk-resident) program that reads data from the STDIN, sorts it in a way the user specifies, and then writes the result to the STDOUT.

The MS-DOS manual gives the essentials for using the SORT program, but it is stingy with examples and leaves it to the user to find SORT's limitations.

Let us assume that we have a telephone directory listing created and maintained by our editor in a file named TEL.LST. Each record is on one line; each field starts at a fixed location on the line, starting with the name, followed by the address, and the phone number; each line is terminated with a carriage return.

To sort this file to the screen, all we need do is issue this command:

```
TYPE TEL.LST | SORT
```

The file will then be sorted in ascending alphabetical order. Here, the "|" character is used to command MS-DOS to "pipe" the output of the system-resident TYPE command into the input of the transient SORT program.

The SORT program can do more than just sort listings beginning with the first character in the line. SORT can take a /R

switch and a /+N switch. The /R switch is simple enough; it just reverses the sorting order. But the /+N switch is something else, as we will see.

The /+N switch causes the sort to start with column N within the line. It does nothing to rearrange the characters that make up the line. Suppose all of the ZIP codes in the telephone directory line started in the same column—say column 60. Then we should be able to issue the command:

TYPE TEL.LST | SORT /+60

and sort by ZIPs, right? Not necessarily so—it depends upon the editor used and how the blank spaces in the line were handled.

Some editors save storage space by substituting tabs in place of spaces wherever possible. So a line may take up 80 character spaces on screen or paper, but considerably less in memory or on disk. (Some editors give you the option of deciding whether tabs will be inserted.)

If the file containing the ZIPs was created with PeachText, for instance, the sorting is quite unpredictable. I have read reports that SORT does not work properly with WordStar, either. On the other hand, if the file were created with an editor like MS-DOS's BSE, and if spaces were used to justify the columns, then the file would sort properly by ZIP code.

We all know that we can't be sure of what we read or hear. (So what's new?) The best writers make mistakes; besides, things have a way of changing after the written word is put to print.

Chris DeVoney, in his excellent book, *MS-DOS User's Guide*, explains that the MS-DOS SORT program follows the ASCII character sequence for sorting. He goes on, therefore, to say that "TEXT" will sort ahead of "text".

That assertion does not agree with my findings, and I can find nothing in the MS-DOS manual relating to this subject.

My SORT program, furnished with MS-DOS version 2.11, does not sort as DeVoney describes. My program considers "TEXT", "text", and "TeXt" to be identical; their sequence in the sorted file is on a first-come, first-served basis.

The characters "X" and "x" are not considered a mismatch by the MS-DOS version 2.11 SORT program (even though they are), because this program is not case sensitive. To get a firm idea of the sorting order, you may consult your ASCII chart, appendix B of the Z100 User's Manual.

And bear in mind that SORT compares characters in like positions in the two sorted strings until a mismatch is obtained. For example, the mismatch occurs on the third comparison when "text" is compared with "test".

When it comes to numbers, however, the larger 199 will sort ahead of the smaller 20. This is because, once again, the comparison stops on the first digit that a mismatch occurs; 1 comes before 2, hence 199 comes before 20.

Here are four examples of SORT commands which may be adapted for specific use:

(1) DIR | SORT /+14

This sorts the directory by file size and displays it.

(2) SORT < ONE > TWO

This sorts file ONE and saves it as file TWO.

(3) SORT < ONE > CON

This sorts the file named ONE and displays the sort on the screen.

(4) SORT < ONE

This is the same as (3) above. (CON is the STDOUT.)

Finding a string somewhere

I'll start this section, which relates to the FIND filter, with a challenge.

Can you list a large directory of mixed files and exclude from the listing all files that have the .COM extension? In other words, can you list a directory that is the complement of the familiar DIR *.COM directory listing?

If you can do that right off, then go directly to the head of the class, and proceed to the next section.

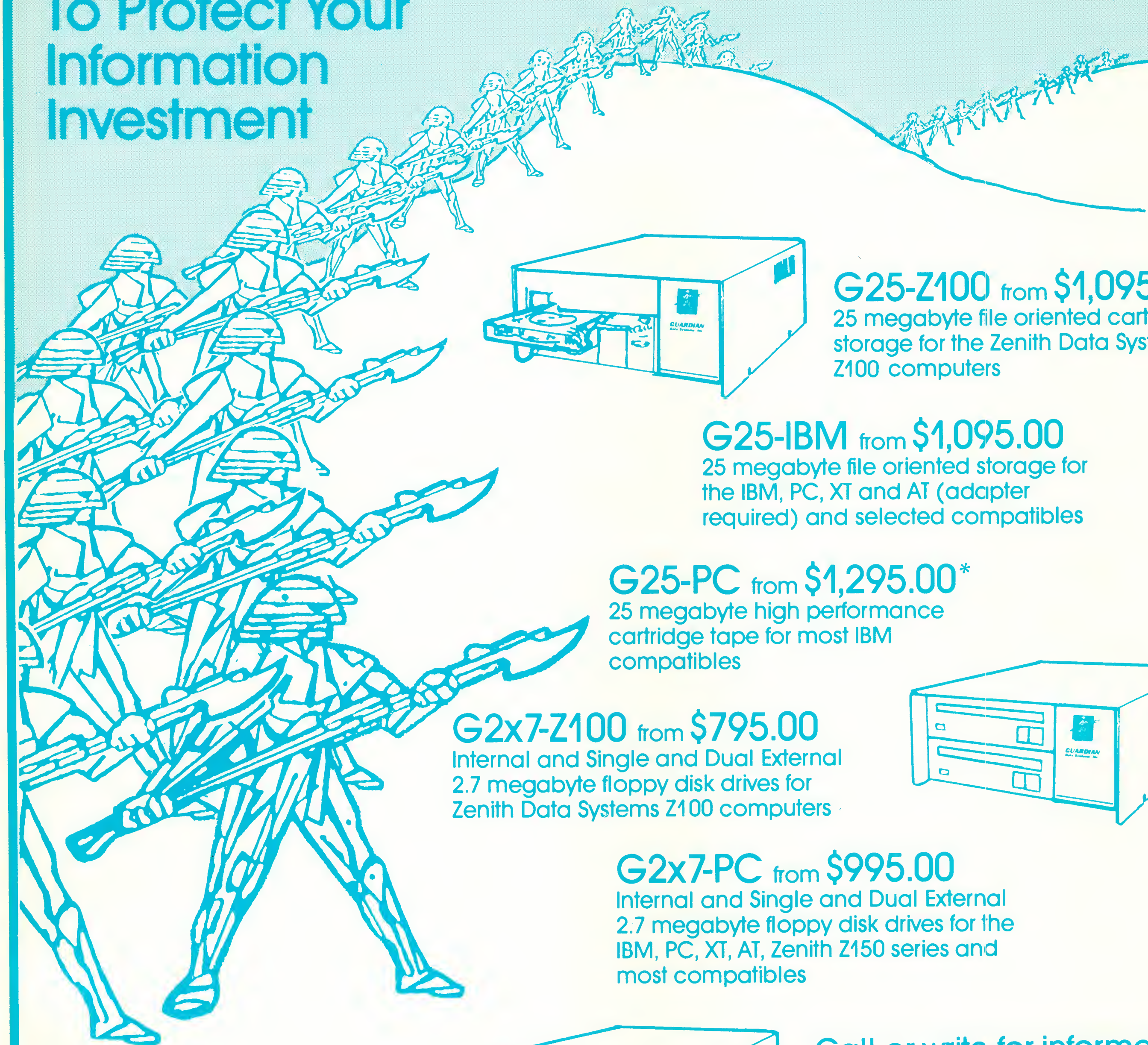
The program FIND.EXE is found on the MS-DOS distribution disk. FIND searches for a specific string in one or more specified files, and does various things once the string is found. (Recall that a string is a series of printable characters taken together and included between quotation marks—"this is a string".)

The FIND program reads the text file(s) that the user specifies; it normally displays the lines of the searched file that contain the search string. The FIND program provides the user with a selection of four switches: /N, /V, /I, and /C. These switches give versatility to the program, as we shall soon see.

The /N switch displays line numbers

THE GUARDIANS ARE GATHERING

To Protect Your Information Investment



G25-Z100 from **$1,095.00**
25 megabyte file oriented cartridge storage for the Zenith Data Systems Z100 computers

G25-IBM from **$1,095.00**
25 megabyte file oriented storage for the IBM, PC, XT and AT (adapter required) and selected compatibles

G25-PC from **$1,295.00***
25 megabyte high performance cartridge tape for most IBM compatibles

G2x7-Z100 from **$795.00**
Internal and Single and Dual External 2.7 megabyte floppy disk drives for Zenith Data Systems Z100 computers

G2x7-PC from **$995.00**
Internal and Single and Dual External 2.7 megabyte floppy disk drives for the IBM, PC, XT, AT, Zenith Z150 series and most compatibles

10 to 40 Meg Hard Disk Upgrades for Zenith Data Systems Z150 and IBM Compatibles

*Available Soon

Call or write for information and the name of your nearest Guardian Data Systems Dealer.

GUARDIAN
Data Systems, Inc.

Distributed by
SYSTEMS INNOVATIONS INC.
44 Stedman Street
Lowell, Massachusetts 01851

HOURS: 9 AM-6 PM MON.-FRI.
THURS. TIL 8 PM • SAT. 10 AM-3 PM

(617) 459-4449

"Your Complete Computer Resource Center"

Circle #106 on Reader Service Card

```
TYPE DIR.LST | SORT /+60

DIR | FIND /V "BAT"

DIR | FIND "84" | SORT /+14 > 1984.FIL

FIND /N "pipe" A:FILTER.DOC B:PIPE.DOC

FIND "filter" BIG.DOC

MORE < THIS.FIL

SORT < THIS.FIL > THAT.FIL

DIR C: | SORT /+9 | MORE

FIND "TIME" B:TIMESORT.BAS

FIND /V/N "ZQ" TEXT.FIL

TYPE A:INDEX.DOC | FIND " specific name"
```

Table 1. Some sample MS-DOS command lines which employ filters. Based on the ideas discussed in the text of the article, these could be modified to accomplish tasks of your own.

and the line containing the searched string. For example, let's say I issued the command:

FIND /N "switch" THIS.FIL

Then THIS.FIL would be searched for the word "switch". When "switch" was found in the preceding paragraph, the display would look like this:

"[164] The /N switch displays ....

The /V (verbose) switch causes FIND to display every line in the searched document that does *not* contain the searched string. This brings us to the challenge question I asked. To display all the files in a directory that do not have the .COM file extension, issue this command:

DIR | FIND /V "COM"

If you have the occasion to eliminate the lines in a text that contain an offensive word, then FIND used with the /V switch will do the job.

The FIND program is case sensitive as far as the search string is concerned. That is, if we searched for "switch", we would not find "Switch", unless we used the /I switch. When the /I switch is used, FIND will ignore the upper/lowercase distinction for letters in the search string. (This is the way that the SORT filter works on my MS-DOS disk.)

The /C switch provides a useful way to count the number of times a given word is used in a text file. Enter the command:

FIND /C "word" BIG.FIL

and BIG.FIL will be searched for "word". Typically, you could receive a screen display that looks like this:

---------- BIG.FIL 5

Here, 5 is the number of times that "word" appeared in the BIG.FIL file.

Have you ever written a BASIC program that had several hundred lines in it, and then found it necessary to change a variable name? Where else in the program did this same variable appear?

Without an editor, this could result in a tedious search. It's much easier to use this command:

FIND "COST" FINANCE.BAS

BASIC already has line numbers, and each line containing the word COST would be displayed on the screen. This would make it easy to find all of the places the COST variable appeared in your BASIC program.

Summary of pipes, redirection, and filters

The MS-DOS filters, when used with the piping and redirection attributes of MS-DOS, provide considerable power; unfortunately, good, practical applications are not so easy to create. It seems that we have a tool waiting for a job to do. The examples I have seen in the MS-DOS manual deal more with the syntax of the commands than with practical applications.

Be that as it may, in Table 1 I am offering a series of examples that work, and perhaps you can modify them to do some specific task you require. These examples are offered without explanation, since the principles involved were covered in the preceding paragraphs. Each command is to be issued from the command line.

And so I close this summary, and the "Z100 Notebook" for this issue. Enjoy!

Additional Information

The C User's Group
Box 97
McPherson, KS 67460
316/241-1065

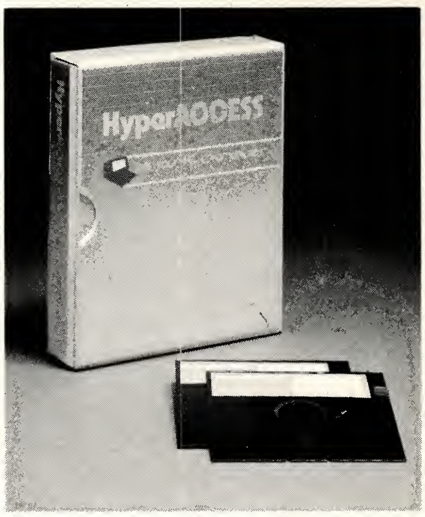
The Software Toolworks
14478 Glorietta Drive
Sherman Oaks, CA 91423
818/986-4885
(Toolworks C)

C Ware Corporation
P.O. Box C
Sunnyvale, CA 94087
408/720-9696
(DeSmet C)

J. E. Hendrix
Box 8378
University, MS 38677
(author of Small-C)

Supplier Notes

MS-DOS Communications Program Eases Learning and Use



HyperAccess is a new communications program from Hilgraeve Inc. Like its predecessor ACCESS, it is designed to ease access to remote computer systems.

Among HyperAccess's features is a startup program for first-time users. This startup program handles initial setup, and also provides an on-screen tutorial which simulates use of CompuServe and other systems. The first time you use HyperAccess, the program memorizes your password and user ID. It then enters them for you in subsequent calls.

HyperAccess also lets you store more than 100 remote systems' names, phone numbers, settings, and automatic sign-on sequences. It comes ready for use on CompuServe, The Source, MCI Mail, EasyLink, and other bulletin board systems.

You can allow your computer to be run from remote micros or terminals, by defining up to 100 passwords. HyperAccess can also be set to let each remote user read or type messages, transfer files, run a program, or have full access to the operating system.

HyperAccess boasts a clear, comprehensive language which may be used for automating routine steps, making unattended calls, or creating custom functions, prompts, and menus.

The program is capable of communicating with almost any mainframe, mini, or micro, because it emulates most terminals. The list of these terminals includes the VT-52, VT-100, VT-102, TeleVideo 900 series, IBM 3101, and the H19. It also transfers files using XMODEM, Kermit, or other methods.

According to Hilgraeve, HyperAccess will work with any modem, including 2400-baud models and the U.S. Robotics S100 modem. It is available for the Z150 and PC-compatibles, and will also work with the Z100, '110, and '120 with or without Gemini or Easy PC boards. HyperAccess retails for $149. Registered ACCESS owners may upgrade by paying the difference between the prices of the two programs. For details, contact Hilgraeve Inc., P.O. Box 941, Monroe, MI 48161, 313/243-0576.

BASIC Compiler for MS-DOS Systems Supports 8087

MT8087 is a BASIC compiler designed for use on MS-DOS systems with either an 8087 or an 80287 co-processor. The compiler, distributed by Softaid, Inc., is able to perform numerical computations 30 to 40 times faster than Softaid's MTBASIC. (Most MTBASIC programs will compile in a second or two and will then run about 50 times faster than interpreted BASIC code.)

Other features of MT8087 include 18 digits of accuracy and a range of 10^{-308} to 10^{308}. Trig, log, and exponential functions are computed with speed and accuracy.

Debugging programs is easier than usual with MT8087. MT8087 is a compiler that acts like a fast interpreter since each line's syntax is checked as you enter a BASIC program. A single command compiles the program to object code and executes it. The code generated by MT8087 can be stored to disk files and executed stand-alone. Code stored to disk can then be burned into your target system's ROM. No runtime license is required to distribute the object code.

For more information, contact Softaid, Inc., P.O. Box 2412, Columbia, MD 21045, 301/792-8096. MT8087 retails for $79.95 plus $3.50 for shipping and handling, $10 for overseas orders. Maryland residents include 5% sales tax.

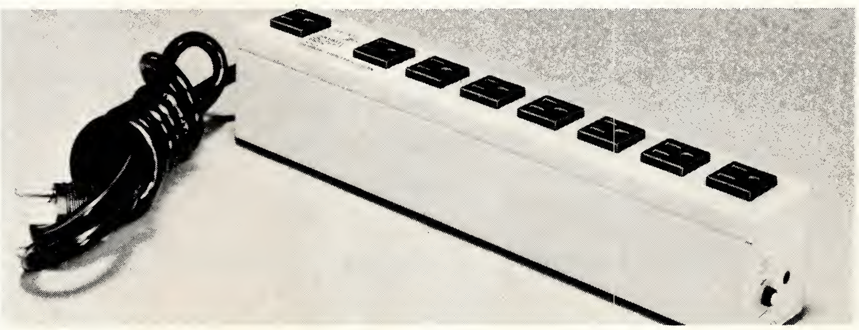
Outlet Box Provides Surge Protection, Single-Switch Control

Heath Company has introduced the GD-1495 Smart Outlet Box—a device which allows single-switch control of seven AC outlets and provides protection from line surges and transient spikes.

One of the seven outlets on the box is designated the control socket. Turning on any device plugged into this control socket will cause power to be sent to the six other AC outlets. A separate unswitched socket is intended for a clock or other device which may require a constant AC source.

The Smart Outlet Box also provides surge and noise protection. It is a UL-approved surge protector and a UL-approved power tap. It is designed with a new surge absorber which opens under direct surge conditions. This eliminates the "hot-to-chassis" ground problems which can result from the usual MOV surge suppressor.

The GD-1495 Smart Outlet Box is available in kit ($59.95) or assembled ($99.95) form from Heath Company, Benton Harbor, MI 49022. For ordering information, call 616/982-3275.



Classified Ads

Hardware

FOR SALE—Z67 10 meg Winchester/8-inch floppy. $500 or trade? George Potter, P.O. Box 88043, Seattle, WA 98188, 206/228-7433.

H/Z89 w/64K, one H17 drive, software, all manuals. Mint condition. $750 or best offer. P. Davis, 307 Jewel Ave., St. Charles, IL 60174, 312/584-7099.

H89, Z37, 64K, two SS/DD drives. Software. $800. Mike Ulis, 337 Cody Road, San Dimas, CA 91773. 818/579-8687.

UPGRADE YOUR HEATH/ZENITH TO 640K. Dolen's $89.95 G-2 user-installed option upgrades 2960 memory boards to full 640K for model 150 or 160. G-3 factory-installed memory expansion is only $139.95. No additional board is required. Prices include shipping. Call 203/222-1174 to order. MasterCard and Visa welcome. *Dolen Computer Corporation*, 10 Bay St., Westport, CT 06880.

ZW110-32 LOW PROFILE COMPUTER, 11-megabyte hard disk, Z-216 (8087 coprocessor—with software to use it), CDR speed module, 768K RAM, ZVM-135 hi-res RGB monitor, and extensive software collection. $4,350 or best offer. Anthony Herbst, 2451 McIntosh Way, Maitland, FL 32751-4004, 305/628-0755.

To get in touch with your fellow Heath/Zenith users, place your short notice in *Sextant*'s classified section. The rate is only 75¢ per word, with a minimum of 15 words. Send your typewritten ad and payment to: Sextant, Classified Ad Department, 716 E Street, S.E., Washington, DC 20003. Please include your name and phone number for our records. Deadline is March 19 for the May-June issue, and May 21 for the July-August issue.

Circle #124 on Reader Service Card

Software

AT LAST! A complete set of graphics routines for Turbo Pascal and your Z100. Provides the graphics support that PC users get from Turbo 3.0 plus many enhancements for the Z100. Pascal and assembly source code included! Requires: 64K color RAM, Turbo 2 or 3, MS-DOS. Price $29.95. Zimbazi Products, 7405 Berkman Drive, Austin, TX 78752.

C. ITOH PROWRITER graphics characters for WatchWord on Z100. Create printable outlines, borders, block graphics. Not a screen dump. Price $10. Bill Codington, 196 Hillside St., Boston, MA 02120.

IF YOU USE SWEEP, try MTL. Multi-column, paging directory, reverse video tags, multi-level menus. Comprehensive manual. $25 for CP/M-80 or CP/M-86; $35 for both versions in one order. Maryland, 5% tax. Specify computer, operating system, and terminal. Outside U.S.A. add $10, U.S. dollars, U.S. bank. Order or info from: MECART, Inc., P.O. Box 3298, Annapolis, MD 21403.

POWERFUL, FLEXIBLE, XY-PLOTTING with LGRAPH. Featuring color graphics, log, linear scales, point symbols, multivariable plots, label and legend control, zoom mode, linear and nonlinear curvefits, built-in curve editor, more. Plots from disk and saves entries. For Z100, Z150, IBM PC under Z-DOS, MS-DOS. Price $35. Free IFR Flight Simulator with order. DMC, P.O. Box 1547, Kent, WA 98032.

Wanted to Buy

H89/Z90 SYSTEMS, 64K and soft-sector controller (preferred). Please contact Dave at Generic Computer Products, Inc., P.O. Box 790, Marquette, MI 49855 or telephone 906/249-9801.

Books

HOW TO USE: Zenith/Heath Computers is both an introduction to computing for the novice and a reference tool for the expert. It's available for $19.95 (Visa, MC, or check) from Sextant Publishing Co., Dept. S21, 716 E Street, S.E., Washington, DC 20003, 202/544-0900. Add $2.50 *per order* for U.S. shipment and $4 *per book* overseas.

Scuttlebutt

Zenith's **Acoustic Touch System** is a touch-based peripheral, which not only determines X and Y coordinates, but also can detect up to **16 levels of finger pressure**. The system works by measuring the interruption in a matrix of ultrahigh-frequency, inaudible sound waves. Finger pressure could be used to control scrolling speed or to control the zoom function in computer-aided design and manufacturing systems.

Zenith has applied for a patent on the device, which costs $150 per unit in quantities of 2,000 or more.

Zenith Data Systems has signed a contract to provide the **Department of Health and Human Services** with Z200s. The contract is worth $4 million. (For more on the Z200, see page 29 of this issue.)

However, at this writing it looks as if Zenith has **lost** the hotly contested **Internal Revenue Service laptop contract** to IBM.

A new HERO from Heath: HERO 2000 is faster, more dextrous, stronger, and more intelligent than his predecessor HERO 1. When he's determined that his battery is low, he looks for his recharger by sending out an infrared light beam. When the beam hits the recharger, the recharger beams back, and HERO announces, **"I found it."** Then he rolls over to it and recharges himself.

Prices are $3,000 for the kit, $4,500 for a wired version. (We're wondering what the 1,998 models between this and HERO 1 look like.)

Astute readers will note that we're introducing a **new column, and columnist**, this issue. **Walter Janowski** has been with his H89 since 1980. He's an Area Technical Supervisor for Sony Corporation, and a graduate of Valparaiso Technical Institute. He's also done time as an electronics technician, as well as selling and supporting microcomputer products.

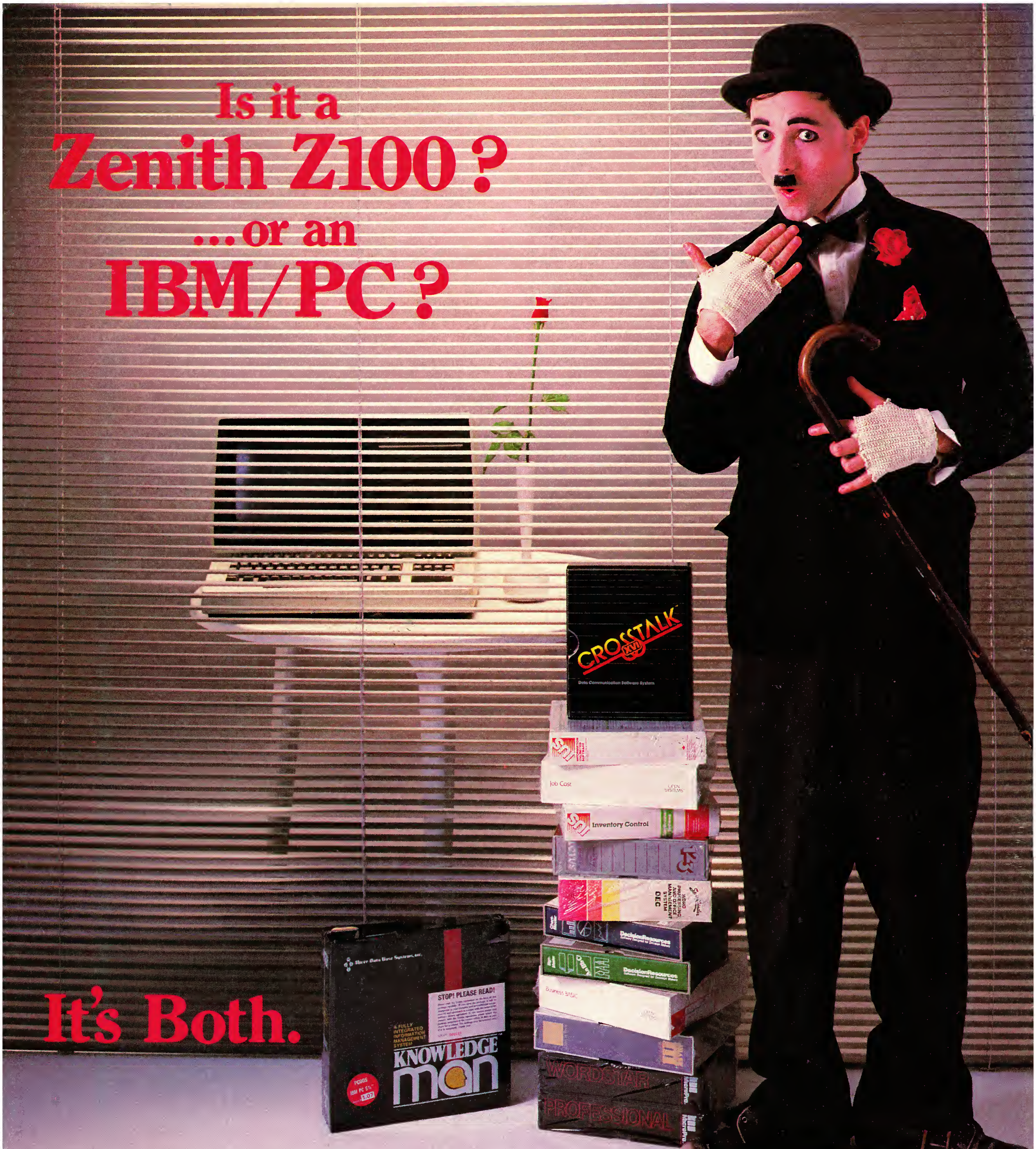
P is for Public Domain.... HUGPBBS is a bulletin board containing all the public domain software submitted over the years to HUG — 3 megabytes of programs for HDOS, CP/M, and Z-DOS/MS-DOS. It's for National HUG members only, and the phone number is 616/982-3956. (To register, follow the instructions in the <I> command. Have your HUG ID ready.)

The First Annual Midwest Heath Users' Group Conference will be held on May 2, 3, and 4 in Bridgeton, Missouri, just two miles from Lambert International Airport. A $5 ticket will get you into the conference meetings, the vendor area, and will give you a chance at a door prize.

You must register in advance to be eligible for the **Grand Prize** drawing at the Saturday night banquet. The banquet ticket costs $20 and also gets you into the meetings and vendor area.

For more information, or to register, write to Midwest HUGCON, Attn: Conference Registration, P.O. Box 482, Collinsville, IL 62234; or leave a note on the St. Louis HUG bulletin board, 314/291-1854.

And life goes on.... The **11th West Coast Computer Faire** will be held April 3-6 at the Moscone Center in San Francisco. For information, call Computer Faire, Inc., 617/965-8350.



Because it's a Zenith Z100 equipped with a Gemini Emulator Board – and that makes it compatible with virtually all 5,000 programs in the IBM/PC library.

It means having the power and flexibility of two excellent computers in one, because a Gemini enhanced Z100 delivers the computing capabilities of both systems.

It instantly selects the proper mode for your existing Zenith programs or for any program from the IBM/PC library.

And what's more, there is no loss of computer power and no loss of time. Just plug in the Gemini Emulator Board and go to work.

Gemini Emulator Boards: available at your local Heathkit Electronics Centre for less than the cost of many popular integrated software packages.

GEMINI
technology inc.

11–11151 Horseshoe Way,
Richmond, B.C. Canada V7A 4S5

For general information or technical support, call (604) 274-1283.

Circle #178 on Reader Service Card

IBM and IBM/PC are registered trademarks of International Business Machines. Gemini is a registered trademark of D.E.L. Professional Systems Inc.